Akraino Technical Community

Introduction

1 Guiding Principles

2 Structure of the Technical Community

2.1 Technical Mission of Akraino Edge Stack

3 Per Project

3.1 Project Roles

3.1.1 Contributor

3.1.2 Committer

3.1.3 Project Technical Leader:

3.1.3.1 Project Technical Leader Candidates

3.1.3.2 Project Technical Leader Voters

3.1.3.3 Project Technical Leader Election Mechanics

3.2 Project Operations

3.2.1 Project Decisions Making Process

3.2.2 Committer Lifecycle

3.2.2.1 Adding Committers

3.2.2.2 Adding Committers to moribund projects

3.2.2.3 Removing Committers

3.3 Project Lifecycle

3.3.1 Akraino Project Lifecycle

3.3.2 Akraino Edge Stack Project Types

3.3.2.1 Akraino Edge Stack Feature Projects (a.k.a Development Project)

3.3.2.2 Akraino Edge Stack Integration Projects (Blueprints)

3.3.2.2.1 Akraino Blueprint Family and Life Cycle of Blueprints

3.3.2.2.2 Akraino Use Cases, blueprint families and blueprints

3.3.2.2.2.1 Template 1 - Use case template

3.3.2.2.2.2 Template 2 - Blueprint family template

3.3.2.2.2.3 Template 3 - Blueprint template

3.3.2.3 Akraino Validation Projects

3.3.2.3.1 Feature project unit testing in Community CI lab

3.2.2.3.2 Blueprint and application testing in Akraino Edge flock lab

3.2.2.3.2.1 Blueprint testing

3.2.2.3.2.2 Application and VNF testing

3.3.3 Project Lifecycle Overview

3.3.4 Project Lifecycle States and Reviews

3.3.5 Tailoring

3.3.6 Reviews & Metrics Overview

3.3.7 Project Reviews

3.3.7.1 Incubation Review

3.3.7.2 Maturity Review

3.3.7.3 Core Review

3.3.7.4 Termination Review

3.3.8 Mature Release Process

3.3.8.1 Release Plan

3.3.8.2 Release Review

3.4 Amendments to the Technical Community Document

4 Technical Steering Committee

4.1 Akraino Community Active Contributors

4.2 TSC Members

4.3 TSC Functional Roles

4.3.1 TSC Chair and Co-Chair

4.3.1.1 Responsibilities

4.3.3 Coordinators

4.3.3.1 Coordinator Description

4.3.3.2 Coordinator origin.

4.3.3.3 Coordinator and coordination area lifecycle

4.4 TSC Operations

4.4.1 TSC Decision Making Process

4.4.2 TSC Election Candidate and Voter Eligibility

4.4.2.1 TSC Members

4.4.2.1.1 Candidate and Voter Eligibility

4.4.2.2 TSC Chair and Co-Chair functional roles

4.4.2.2.1 Candidate and Voter Eligibility

4.4.2.3 TSC Coordinators functional roles

4.4.2.3.1 Candidate and Voter Eligibility

4.4.3 TSC Election Mechanics

4.4.3.1 TSC Member Elections

4.2.3.2 TSC Member Election Mechanics

4.4.3.2.1 Enforcement of organization TSC member limits.

4.4.3.2.2 Interim elections

4.4.3.3 TSC Chair/Co-Chair/Coordinator Election Mechanics

4.5 Responsibilities of the TSC.

4.6 TSC Subcommittees

4.6.1 Membership

4.6.1.1 Subcommittee Membership Eligibility

4.6.1.2 Subcommittee Chair / Vice Chair

4.6.1.3 Subcommittee Chair / Co-Chair Elections

4.6.1.4 Subcommittee Voter Eligibility

4.6.1.5 Subcommittee Election Confirmation

4.6.2 Advisory role

4.6.3 TSC subcommittee lifecycle.

4.6.3.1 Creation of a TSC subcommittee

4.6.3.2 Update of a TSC subcommittee

4.6.3.3 Conclusion of a TSC subcommittee

4.6.4 Subcommittee vs. coordinator

Glossary

Introduction

The Akraino Edge Stack technical project has been established as Akraino Project a Series of LF Projects, LLC ("Akraino Edge Stack" or, alternatively, the "Project"). LF Projects, LLC is a Delaware series limited liability company. Governance for the Project is detailed within the Project Technical Charter available at akraino.org ("Technical Charter"). This Technical Community Document is intended to provide additional operational guidelines for the Project, and is subject to the Technical Charter. "Akraino" is the term used in this document that refers to "Akraino Edge Stack" (full form).

1 Guiding Principles

The Akraino Edge Stack Project a Series of LF Projects, LLC ("Akraino") will operate transparently, openly, collaboratively, and ethically. Project proposals, timelines, and status must not merely be open, but also easily visible to outsiders.

2 Structure of the Technical Community

The Technical Community consists of multiple projects and a Technical Steering Committee that spans across all projects.

2.1 Technical Mission of Akraino Edge Stack

The Akraino Edge Stack Technical mission to focus on following areas

1. Create end to end configuration for a particular Edge Use case which is complete, tested and production deployable meeting the use case characteristics {Integration Projects - Blueprints}.

   Production deployment means the blueprint has passed unit and integration testing and meets the blueprint's use case characteristics.

2. Develop projects to support such end to end configuration. Leverage upstream community work as much as possible to avoid duplication. {Feature Projects}
3. Work with broader edge communities to standardize edge apis {Upstream Open Source Community Coordination - For example, Socialization, so community tools and Blueprints can interoperate. This work can be a combination of an upstream collaboration and development within the Akraino community [i.e. a feature project]}
4. Encourage Vendors and other communities to validate Edge applications and Virtual Network Functions on top of Akraino blueprints {Validation Project - ensures the working of a Blueprint}

In order to have focused work in support of initial releases, the Akraino community preference is to support Edge Blueprints related to enterprise and industrial IoT, and carrier edge network use cases. Board has the authority to define, modify, prioritize any additional industry sectors that need to be supported by the Akraino Edge Stack releases.

The Edge cloud stack placement could vary between Telco Offices to Customer premise and anything in between. Akraino Edge Stack community blueprints should be capable to deploy and address different edge cloud placement options.

As an example, the picture below illustrates the enterprise edge and carrier edge network Edge use cases and possible edge placement.

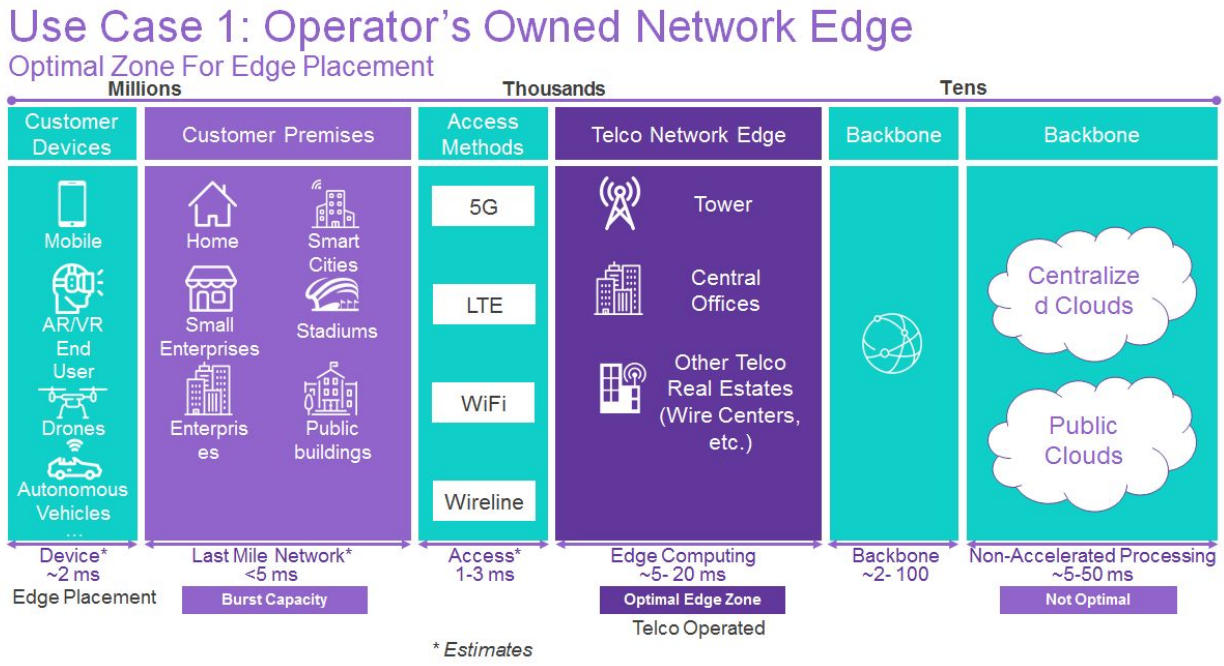

Figure 1 - Sector 1: Carrier edge network edge use case and edge placement

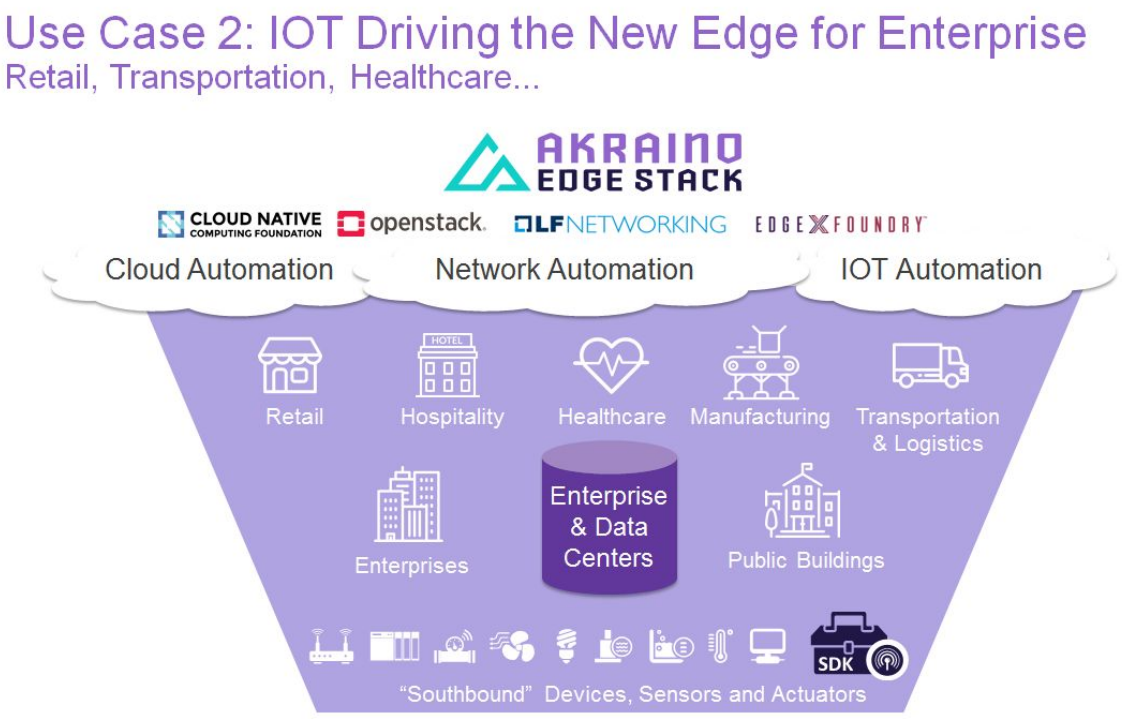

Figure 2 - Sector 2: Enterprise and Industrial IoT use cases

3 Per Project

3.1 Project Roles

3.1.1 Contributor

A Contributor is someone who contributes to a project. Contributions could take the form of code, code reviews,Wiki and documentation contributions, Jira activities or other artifacts. Contributors work with a project's Committer and the project's sub-community. A Contributor may be promoted to a Committer by the project's Committers after demonstrating a history of contributions to that project.

3.1.2 Committer

For each project there is a set of Contributors approved for the right to commit code to the source code management system (the "Committers") for that project.

- Committer rights are per project; being a Committer on one project does not give an individual committer rights on any other project.
- The Committers will be the decision makers on all matters for a project including design, code, patches, and releases for a project.
- Committers are the best available individuals, but usually work full-time on components in active development.
- All project committers information such as name, company, and Contact information should be documented in the wiki under the project.

In order to preserve meritocracy in selection of Committers while ensuring diversity of Committers, each initial project is encouraged to take on at least two Committers from different companies (subject to meritocracy).

3.1.3 Project Technical Leader:

A project is required to elect a Project Technical Leader ("PTL"). The PTL acts as the de facto spokesperson for the project (feature projects and integration projects).

3.1.3.1 Project Technical Leader Candidates

Candidates for the project's Project Technical Leader will be derived from the Committers of the Project.

Candidates must self nominate.

3.1.3.2 Project Technical Leader Voters

Only Committers for a project are eligible to vote for a project's Project Technical Lead.

3.1.3.3 Project Technical Leader Election Mechanics

An election for Project Technical Leader occurs when any of the following are true:

- The project is initially created
- The Project Technical Leader resigns his or her post
- The majority of committers on a project vote to call a new election
- One year has passed since the last Project Technical Leader election for that project

3.2 Project Operations

3.2.1 Project Decisions Making Process

Technical and release decisions for a project should be made by consensus of that project's Committers. If consensus cannot be reached, decisions are taken by majority vote of a project's Committers. Committers may, by majority vote, delegate (or revoke delegation) of any portion of such decisions to an alternate open, documented (wiki), and traceable decision making process.

3.2.2 Committer Lifecycle

3.2.2.1 Adding Committers

- Initial Committers for a project will be specified at project creation
- Committer rights for a project are earned via contribution and community trust. Committers for a project select and vote for new Committers for that project.
- New Committers for a project should have a demonstrable established history of meritocratic contributions.

3.2.2.2 Adding Committers to moribund projects

In the event that a project has no active committers (e.g., due to resignations, etc.), the TSC may appoint an interim Committer from a project's active Contributors. This term shall last until the next release date, after which time the Committer must stand for election from amongst other Committers on the project to maintain his or her status. In this special case, approval requires a majority of committers who respond within two weeks. If no one responds by the deadline, then the committer status is

approved. This provision allows a project to continue development following an unexpected change in personnel.

The method by which the TSC appoints an interim Committer is first by request to the Akraino-TSC email list indicating the request to appoint an interim Committer for a project. After the reception of such an email, the normal TSC decision process applies.

3.2.2.3 Removing Committers

A Committer may voluntarily resign from a project by making a public request to the PTL to resign (via the project email list and cc to tsc@lists.akraino.org).

A Committer for a project who is disruptive, or has been inactive on that project for an extended period (e.g., six or more months) may have his or her Committer status revoked by the project's Project Technical Leader or by 2/3 supermajority vote of the project's committers.

The Project Technical Leader is responsible for informing the Technical Steering Committee (TSC) of any committers who are removed or resigned via the email list: tsc@lists.akraino.org.

Former committers removed for reasons other than being disruptive may be listed as 'Emeritus Committers'. That title expresses gratitude for their service, but conveys none of the privileges of being a Committer.

3.3 Project Lifecycle

3.3.1 Akraino Project Lifecycle

The activities of the Akraino community are articulated around projects and releases. The scope of each project is aligned with Akraino's charter and the scope of each release is defined with the objective to fulfill a particular Edge use case(s).

A project is a long term endeavor setup to deliver features across multiple releases, which have a shorter lifespan. The project and release lifecycle should provide sufficient visibility to all community members and allow teams to coordinate with one another.

This document covers the Akraino project lifecycle. The Release Lifecycle will be documented in the wiki.

3.3.2 Akraino Edge Stack Project Types

Akraino will support three categories of projects and upstream coordination activities related to the projects as shown in Figure 3 and further illustration are available under the section 3.3.2.1, 3.3.2.2 and 3.3.2.3 of this document.

The Akraino Edge Stack Community goal is to deliver fully integrated and production deployable solution for the users. The combination of Feature projects (developing missing edge functionalities on non-upstream open source components), Integration projects (integrated end to end stack) and End To End (ETE) validation projects of the blueprints along with edge applications running on top of it, delivers the productional deployable solution needed by the industry.

Akraino community uses upstream-first principle to avoid technical debt on the upstream open source components: Akraino projects should not carry patches against upstream projects, but collaborate with

and contribute designs and patches to the respective upstream projects to address gaps. Exceptions must be approved by the TSC.

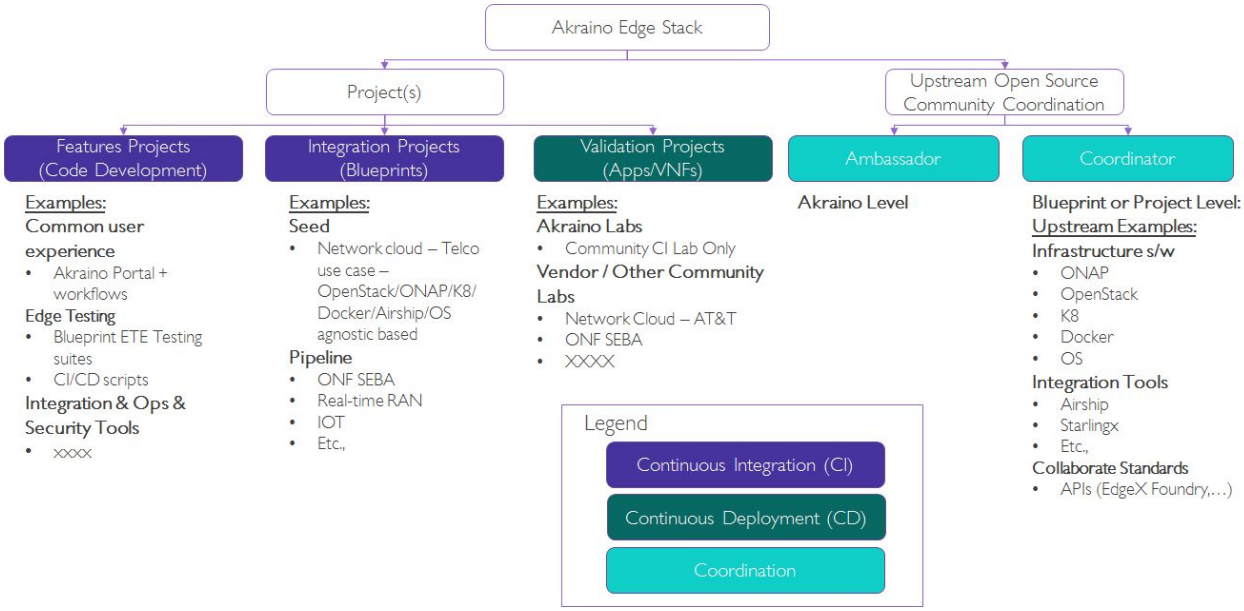

Figure 3 – Akraino Edge Stack Project Types & Scope

3.3.2.1 Akraino Edge Stack Feature Projects (a.k.a Development Project)

The Akraino Edge Stack Feature projects develop edge features, functionalities, interfaces and modules required by an Akraino blueprint or multiple Akraino blueprints to satisfy the Edge use cases and requirements. Feature projects focus on the development work required by Akraino community instead of upstream open source components.

Examples of Feature Projects include Akraino Portal and Workflows to support common user experience, Blueprint testing modules or suites, Operational tools, Edge SDKs and APIs, etc..

It is a long term endeavor set up to deliver features across multiple releases, which have a shorter lifespan.

The feature project should address the functionality needed by Akraino specified use cases. During the feature project creation the scope should define whether it will support a single or multiple blueprints. Illustration is shown on Figure 4.

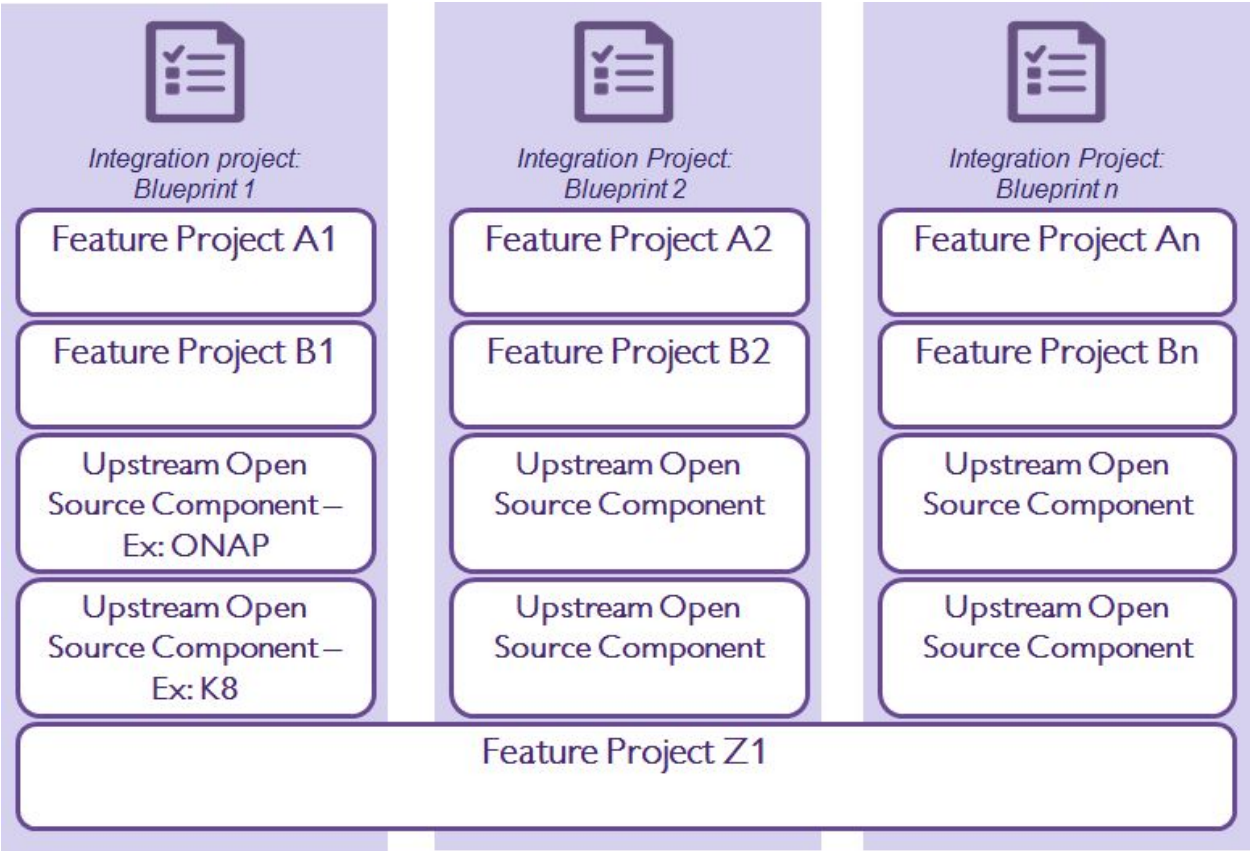

Figure 4 – Relationship between Feature Project, Upstream component and Integration project

3.3.2.2 Akraino Edge Stack Integration Projects (Blueprints)

Akraino Integration Projects create or modify Akraino Blueprints to support specific edge use cases.

A Blueprint is a declarative configuration of an edge Cloud Stack that includes a combination of infrastructure hardware components, software components, Point of Delivery (POD) options, tools to manage the life cycle and CI/CD to manage the blueprint's code. The blueprint should be production deployable to support one or more Edge applications or Virtual Network Functions (VNFs).

A Blueprint family is a collection of blueprints that share common characteristics and numerous POD types. For example, Network Cloud is a family of blueprints to support any Telco Virtual Network Functions (VNFs). Network Cloud - Unicycle A is a blueprint supports Unicycle POD and Network Cloud - Rover A is a blueprint which delivers single server POD.

Akraino Blueprints must include the description of the Edge use cases that articulates the desired business outcome, workload characteristics, design constraints, and operational cost ranges.

The Blueprint can either develop or use the upstream lifecycle management (LCM) tools and deployment automation to support the installation of the Blueprints. All Blueprints need to be tested by the Akraino community to prove that Edge applications and/or VNFs can operate effectively on the Blueprint as requested in the use case .

Community members will use a template for use case characteristics, blueprint family and blueprint template approved by the TSC to describe the hardware, software, deployment configurations, workload characteristics when requesting the creation or modification of an Akraino Blueprint. The requestor should demonstrate the following aspects to the TSC:

1. Each initial blueprint is encouraged to take on at least two Committers from different companies
2. Complete use case and use case characteristics template
3. A lab with exact configuration required by the blueprint to connect with Akraino CI and demonstrate CD. User should demonstrate either an existing lab or the funding and commitment to build the needed configuration.
4. Blueprint is aligned with the Akriano Edge Stack Charter
5. Blueprint code that will be developed and used with Akraino repository should use only Open Source software components either from upstream or Akriano projects.
6. For new blueprints submission, the submitter should review existing blueprints and ensure it is not a duplicate blueprint and explain how the submission differs . The functional fit of an existing blueprint for a use case does not prevent an additional blueprint being submitted.

After TSC review and approval of community member request, an integration Project will be launched to develop the Blueprint. In some cases, community may decide to support single or multiple family of blueprints for a use case depending upon the need and interest (Subject to TSC approval). A blueprint or blueprint family may address one or more Edge use cases.

TSC will decide which blueprints or blueprints family to include into a release based on the maturity status of an individual blueprint within a Blueprint family. Maturity of a blueprint is demonstrated with full CD deployment using a validation project and should have addressed the use case for which the blueprint was created. A release could support multiple blueprints. It is a long term endeavor set up to deliver a blueprint across multiple releases, which have a shorter lifespan.

Blueprint Specifications ultimately define the declarative configuration for each deployment model or Point of Delivery (POD) of a Blueprint. The Point of Delivery (POD) defines the method in which a blueprint is deployed to an edge site. PODs organize edge devices for deployment and enable a cookie-cutter approach to large scale deployments (e.g., 10,000 plus locations) at a reduced cost. For example, an edge location could have a single server or multiple servers in one or more racks. Blueprint families use YAML or similar configuration files. The use of YAML files with different manifest file contents will allow for different configurations within the same Akraino blueprint family.

The below diagram (Figure 5) illustrates the process.

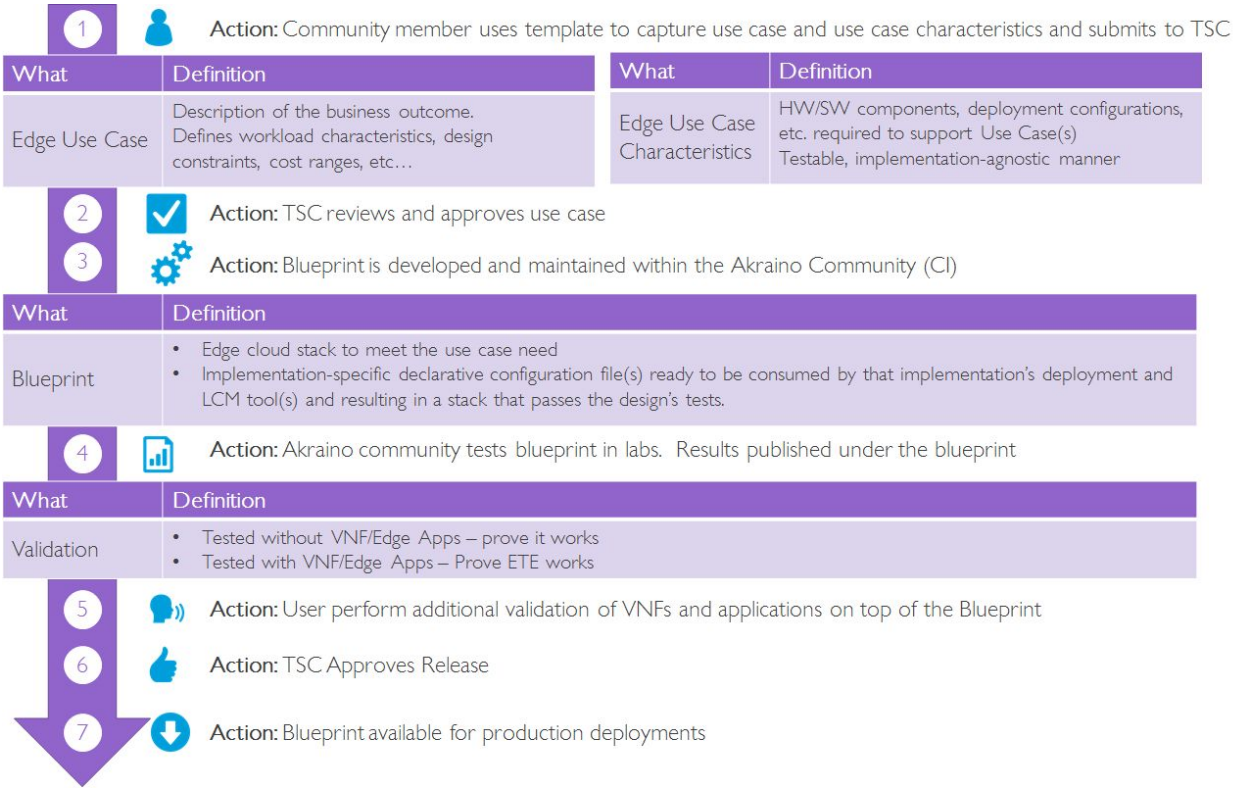

Figure 5 – Akraino Blueprint Creation Process

3.3.2.2.1 Akraino Blueprint Family and Life Cycle of Blueprints

At the Family level, blueprints are differentiated by high level technical attributes which are immutable. Removing or inserting one of these attributes would change any pod delivered by the blueprint to such an extent that it could not reasonably be considered part of the same blueprint family.

For example if a given blueprint family has a Family attribute of deploying a kubernetes based undercloud in a given Akraino release then any pod delivered by that blueprint must also do so.

The decision as to what technical attributes should define the blueprint family will be specific to each blueprint family (and release) and be approved by the TSC.

Each blueprint family should be flexible enough to allow reasonable variation to encourage widespread deployment without requiring the introduction of a new blueprint family. For example, the decision to use Ubuntu or CentOS as the operating system in the Network Cloud blueprint family, should not require a different blueprint family to be introduced instead just different POD configuration (Rover with Ubuntu vs. Rover with CentOS)

The ultimate and final level of classification must be sufficiently definitive to allow any user to reliably deploy a duplicate pod in their own environment. This level of blueprint specification does not prevent a deployable entity (POD) from having a range of possible values (for example Ubuntu 16.X). This ultimate level of technical classification is termed the blueprint Species.

Akraino releases will verify blueprints at this ultimate species level of technical specification to allow reliable and repeated pod replication.

Blueprint taxonomy of technical specification:

→ Family

→ Species (a.k.a blueprint, from which a replica pod can be deployed by anyone)

At the very minimum there would need to be one fully specified and thus deployable pod within a blueprint family (i.e. one species/POD of the blueprint). The number of species within a blueprint family may grow or diminish with Akraino releases.

Note: Specific named examples have been used in this section to help clarify. The use of "Network Cloud" as example only reflects the fact this blueprints had been submitted to Akraino when written.

A use case may be supported by one or more blueprint families i.e. there is not a 1:1 mapping from an Akraino use case to Blueprint families. Likewise a given Akraino blueprint may support multiple Akraino use cases. An example of a Blueprint family are Network Cloud.

Below picture illustrates the Network Cloud blueprint family and POD level specification which allows possibility of different component level.

Blueprint Family	*Blueprint*			
	Name	*POD Type*	*Components / Stack*	*POD Declarative Configuration*
Family: Network Cloud	Network Cloud – Unicycle A	Unicycle	Ubuntu/OS/Neutron based	{yaml files UA}
	Network Cloud – Unicycle B	Unicycle	Centos/OS/ODL based	{yaml files UB}
	Network Cloud – Tricycle A	Tricycle	X	{yaml files TA}
	Network Cloud – Rover A	Rover	A	{yaml files RA}

Figure 6 - Illustration of blueprint family and its blueprints

3.3.2.2.2 Akraino Use Cases, blueprint families and blueprints

Akraino use cases are business driven and must include a clear description of business requirements, operational considerations (cost, user interface, scale, power restrictions, etc…) , and applications expected to operate on the Blueprint. Any community member can submit a use case for review by TSC for further development.

The submitter should use the appropriate templates as described below for the creation or modification of blueprint families or blueprints for TSC review.

Use case template and blueprint family templates are required when requesting creation or modification of a blueprint family.

Blueprint templates are required to create or modify blueprints within a blueprint family. In some cases, new or modification to a blueprint may require updates to existing use case templates and/or blueprint family templates and the submitter should make changes to all 3 templates as required.

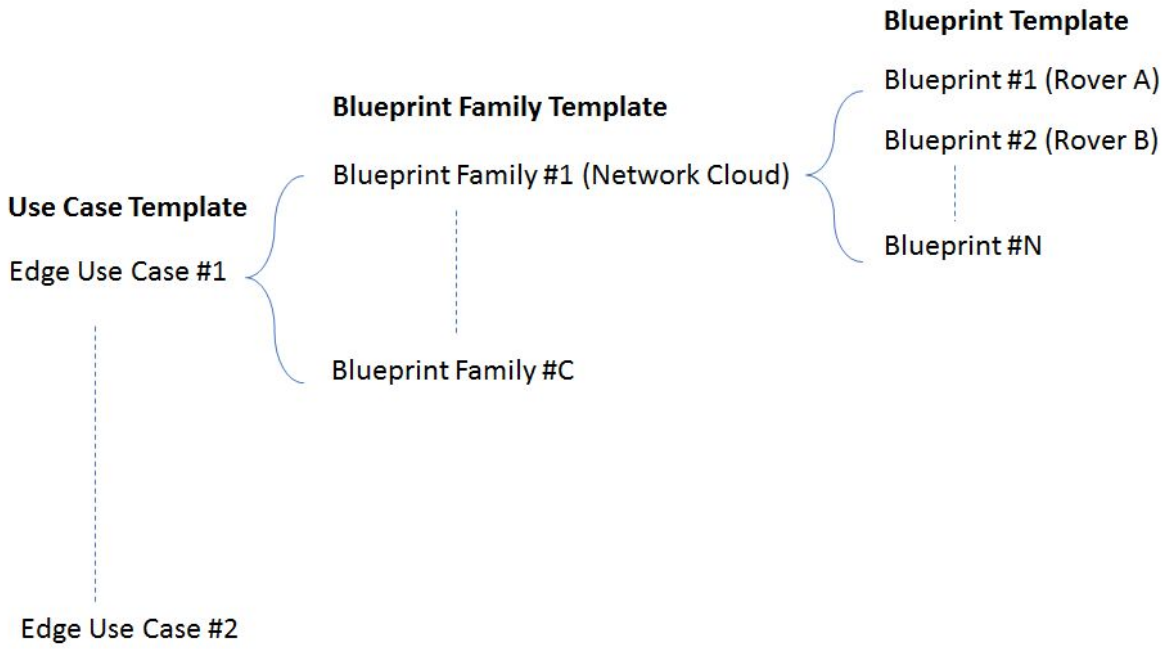

Figure 7 - Templates needed for blueprint family and blueprint creation and modification

When one or more fully defined blueprint addresses the same use case common test cases should be adopted where possible.

Below are the sample templates of a target family of blueprints and the full template will be maintained on the wiki. Certain attributes shall be defined as informational only. For example cost targets whilst key should not be used to pass or fail the verification of a blueprint

3.3.2.2.2.1 Template 1 - Use case template

Below is a sample Use case template. The full template will be maintained on the Akraino Wiki.

Use Case Attributes	Description	Informational
Type	New or Modification to an existing submission	
Industry Sector	Telco and carrier networks	
Business driver	Emerging technologies such as 5G (vRAN, Core) and associated Edge Services requires Cloud instance deployed at the edge of the provider network to support latency need.	

	Without Edge Cloud, above said services cannot be enabled.	
Business use cases	For Example: 1. Customer Edge deployable at Customer premises such as home, enterprises, offices to support Edge services such as SD-WAN 2. Customer Edge deployable at Public buildings such as Stadiums, Smart cities, etc., to support edge applications for example - Video Analytics, AI/ML based detection, etc., 3. Edge Cloud deployable at Cell tower to support RAN related workload 4. Edge Cloud deployable at Central offices ranging from one rack to multiple racks	
Business Cost - Initial Build	For Example: 1. Edge Cloud deployable at Central offices with single rack should be less than 150K $	
Business Cost - Operational	For Example: 1. Edge Cloud deployable at Central offices with single rack should be less than 100K $ as operating cost per year. 2. In-place upgrade of the Edge cloud should be supported without impacting the availability of the edge applications	
Operational need	For Example: Edge Solution should have role based access controls, Single Pane of Glass control, administrative and User Based GUIs to manage all network cloud family based blueprints. The automation should also support zero touch provisioning and management tools to keep operational cost lower	
Security need	For Example: The solution should have granular access control and should support periodic scanning	
Regulations	For Example: The Edge cloud solution should meet all the industry regulations of data privacy, telco standards (NEBS), etc.,	
Other restrictions	Consider the power restrictions of specific location in the design (example - Customer premise)	

Additional details	The Edge Cloud Solution should be deployable across the globe and should be able to support more than 10,000 locations	

3.3.2.2.2.2 Template 2 - Blueprint family template

Below is a sample Blueprint family template. The full template will be maintained on the Akraino Wiki.

Use Case Attributes	**Description**	**Informational**
Type	New or Modification to an existing submission	
Blueprint Family - Proposed Name	Network Cloud Family	
Use Case	Network Cloud	
Blueprint proposed	Central Office deployments ● Unicycle ● Tricycle ● Cruzer Customer Premise deployments ● Rover	
Initial POD Cost (capex)	Examples Only: ● Rover less than $20k ● Unicycle less than $150k ● Tricycle less than $300k ● Cruzer less than $800k	
Scale	Examples Only: ● Rover - 1 server ● Unicycle - 1 rack ● Tricycle - 3 racks ● Cruzer - 6 racks	
Applications	Any type of Edge Virtual Network Functions	
Power Restrictions	Example Only: ● Cruzer - less than 50k watts	
Preferred Infrastructure orchestration	OpenStack - VM orchestration Docker/K8 - Container Orchestration OS - Linux VNF Orchestration - ONAP Under Cloud Orchestration - Airship	
Additional Details	Submitter to provide additional use case details	

3.3.2.2.2.3 Template 3 - Blueprint species template

Below is a sample Blueprint template defining an Akriano species. The full template will be maintained on the Akraino Wiki.

Use Case Attributes	**Description**	**Informational**
Type	New or Modification to an existing submission	
Blueprint Family - Proposed Name	Network Cloud Family	
Use Case	Network Cloud	
Blueprint proposed Name	Unicycle A	
Initial POD Cost (capex)	Examples Only: ● Unicycle less than $150k	
Scale & Type	● Up to 7 servers ● x86 based	
Applications	5G Core or vRAN (RIC)	
Power Restrictions	Example Only: ● Less than 10Kw	
Infrastructure orchestration	OpenStack Pike or above - VM orchestration Docker 1.13.1 or above / K8 1.10.2 or above- Container Orchestration OS - Ubuntu 16.x VNF Orchestration - ONAP Beijing Under Cloud Orchestration - Airship v1.0	
SDN	SR-IOV & OVS-DPDK	
Workload Type	VMs and Containers	
Additional Details	Submitter to provide additional use case details	

3.3.2.3 Akraino Validation Projects

Multiple labs will be used to validate Akraino projects to ensure production quality of blueprint and/or feature project releases. The Community CI lab is owned by the Akraino community and LF. In addition a number of Edge FLock labs are owned by an Akraino community company, organization or participant.

Comprehensive validation of the blueprint within the community will include testing of the OS layer, the undercloud layer, the upper cloud layer, VNF layer and the application layer. Test results should be

available to the Akraino Edge Stack community for review via automated push to the wiki. Any defects identified should be logged in JIRA and assigned to the responsible party. When defects are identified in upstream components, Akraino coordinators should be assigned defect ownership. Akraino coordinators will work with upstream communities to resolve defects. For defects that do not involve upstream communities, impacted Protect Technical Leaders should be assigned defect ownership.

Akraino Validation Projects will also include testing of Akraino releases.

An automation framework should be used for test procedures and leverage existing test automation suites for upstream projects as required. It is the responsibility of the blueprint submitter to ensure that the Edge Flock and Community CI labs can support comprehensive validation of the blueprint and cover all use case characteristics.

The TSC subcommittees tasked with the setup and structuring of operating procedures for the validation labs shall work to define the licensing requirements for the external labs.

3.3.2.3.1 Feature project unit testing in Community CI lab

The Akraino Community CI labs are owned by the Akraino community and LF. Access is controlled by LF best practices.

TSC will propose a subcommittee to maintain and coordinate the Community CI lab.

The following would be conducted in the Community CI lab:

- Unit testing of Feature projects or
- Integration of Feature projects to a blueprint
- Upstream projects together in the support of Akraino Blueprints.

A full CD that works with the Akraino CI pipeline should be included in the validation project.

3.2.2.3.2 Blueprint and application testing in Akraino Edge Flock lab

Akraino Edge flock labs are owned by an Akraino community company, organization or participant.

The Flock labs can be owned and operated by the supplier of the lab with limited access or no access to the hardware but results of the validation testing need to be shared to the community.

TSC will propose a subcommittee to maintain and coordinate the Flock labs.

Blueprint, application and VNF testing may be performed over a number of Flocks labs concurrently in a coordinated manner as required.

- The Flock labs should be able to connect to Akriano Edge Stack CI hosted by LF to pull in Akraino blueprints for the validation. Necessary Firewalls can be opened by the Akriano Community on request.

- All Flock labs should be available most of the year for the community use and if the lab is not used for validation more than 3 consecutive months then it will be removed from the community list.

- Hardware and Network configuration used within the Flock labs should be declared and information should be documented in the wiki
- All the Flock labs addition, modifications or removable need to be approved by TSC
- All historic results (minimum of 1 year) of blueprint validation, Applications and VNF testing should be maintained in the wiki.

3.2.2.3.2.1 Blueprint testing

The test plan and associated test cases to validate the blueprint will be published on the Akraino wiki and reviewed by blueprint stakeholders.

- The lab configuration should support functional, performance, security, and other test cases for the blueprint and cover the full stack for the blueprint.
- Any license or support beyond the Community supplied software should be funded, owned and operated by the flock supplier

Open source test tools and vendor tools with appropriate configurations should drive test traffic profiles that mimic in-scope use cases. Akraino coordinators will engage upstream open source communities to get access to upstream community labs for the Akraino blueprint test team. The submitter is responsible to establish access to necessary vendor labs.

3.2.2.3.2.2 Application and VNF testing

Akraino blueprint releases ensure that applications and virtual network functions (VNFs) can on-board and operate effectively on the Akraino solution that has been deployed based on the creation or modification of an Akraino Blueprint. Flock labs can be further used for the validation of Edge applications and VNFs.

Application and VNF testing may take a number of forms including performance, scalability, security and usability, resilience etc.

Submission of results to the Akriano community is optional in case of commercially sensitive applications.

3.3.3 Project Lifecycle Overview

The project lifecycle provides the freedom for each team to conduct its project according to their needs, culture and work habits. Thus, the project lifecycle is not prescriptive on how each project operates.

Individual blueprint within a blueprint family shall not be tied to the overall Akraino release schedule (e.g. 6 months). An Akraino release can be composed of 1 to N fully verified blueprints or individual feature projects. As such the number of blueprints and contributing projects within a given Akraino release may vary overtime. Figure 8 show this with two blueprint families, Network Cloud and the imaginary Canis Edge and a number of individual blueprint species within each family.

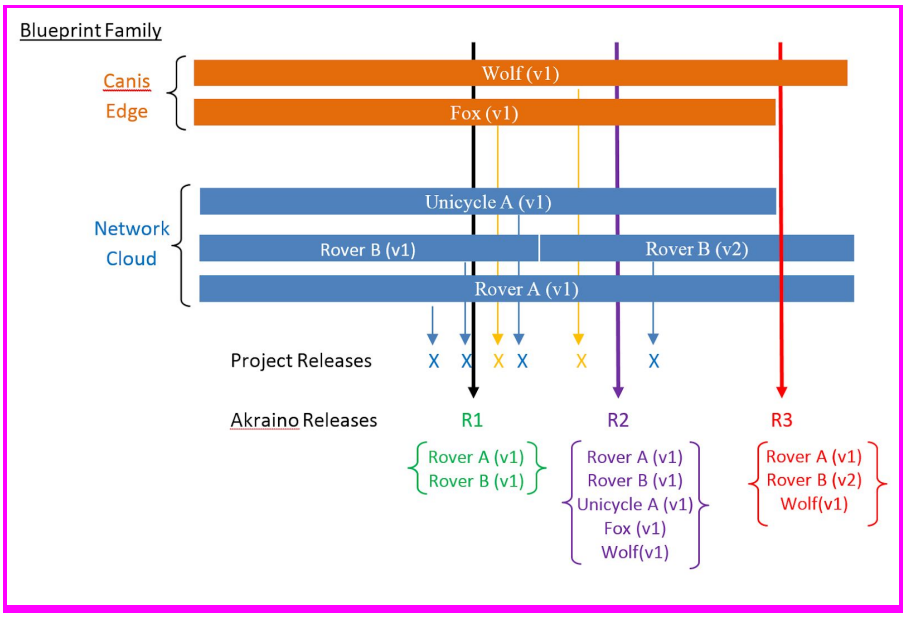

Figure 8 - Project and AKraino release lifecycles

Each project shall provide an expected release plan document published with the blueprint on the wiki. From the project schedules an an overall Akraino release schedule shall be maintained and published.

The project lifecycle process does not impose a duration for the project nor for individual project releases.

3.3.4 Project Lifecycle States and Reviews

Akraino projects' life cycles defines five states that all three project goes through. A project lifecycle may **extend across** multiple project and Akraino releases.
The procedure of moving from one state to the next one is independent from the Akraino release lifecycle and the pace depends on each individual project.

In order to effectively review project progress, **four** reviews are built-in to the project lifecycle.

The lifecycle of a project is depicted on the following diagram:

Project State	**Description**
Proposal	Project doesn't really exist yet, may not have real resources, but is proposed and is expected to be created due to business needs.
Incubation	Project has resources, but is recognized to be in the early stages of development. The outcome is a minimum viable product (MVP) that demonstrates the value of the project and is a useful vehicle for collecting feedback, but is not expected to be used in production environments.

Mature	Project is fully functioning and stable, has achieved successful releases.
Core	Project provides value to and receives interest from a broad audience.
Archived	Project can reach Archived state for multiple reasons. Either project has successfully been completed and its artifacts provide business values, or project has been cancelled for unforeseen reasons (no value anymore, technical, etc.). Project in any state can be Archived through a Termination Review.

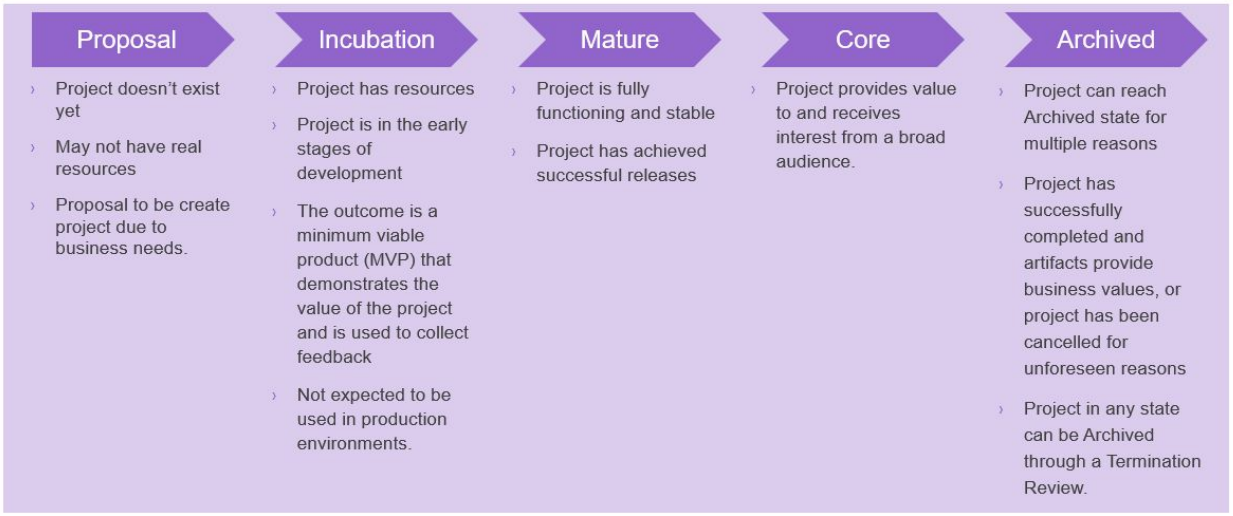

Figure 9 – Akraino Project States

To move from one state to the next state, the Project Team has to formulate a Kick-Off release review to the TSC, by specifying its goal to move up the Project Lifecycle ladder.

From State	To State	Review Description
Null	Proposal	
Proposal	Incubation	Incubation review
Incubation	Mature	Maturity review
Mature	Core	Core review
Core	Archived	Termination review

Note 1: Project proposals are posted in the "Proposed Projects" section of the Akraino wiki. Approved projects are posted to the "Approved Projects" section of the Akraino wiki.

The TSC will review each project proposal request and then vote to approve or reject. Requestors should identify the different Point-of-Delivery options that should be enabled via the Blueprint creation / modification request, explain the continuous integration / continuous deployment (CI/CD) methodology that will be used, explain the test framework that will be used for the Blueprint and define the automation that will be needed to deploy Akraino based on the Blueprint.

Note 2: The proposal submitter can decide to remove projects in "proposal" state that do not progress to incubation state.

3.3.5 Tailoring

A project's release cycle may be tailored by allowing some exceptions to the normal release process. Tailoring may be initiated in **two** ways:

1. By the TSC voting members: TSC voting members reserves the right to allow changes to the process in order to meet criteria that were initially unknown.
2. By Project Team Lead: Any project team lead can email TSC voting members to request tailoring the process for a particular release. The key point in tailoring is to anticipate as much as possible, to justify the request, and document the request in the wiki.

Tailoring practices will be documented as we progress through our releases. The TSC should respond to requests in a timely manner.

3.3.6 Reviews & Metrics Overview

Project promotion across states can only be done by TSC review and voting. During the reviews the candidate projects are evaluated based on predefined metrics and KPIs. The target numbers may vary for each project and state.

- Longevity of the project
- Project follows (or doesn't) the Akraino release cadence
- Requirements have resulted in corresponding implementations
- Comprehensiveness and maturity of the artifacts (code, test cases, documentation) the project produces including contributions/code to partner/upstream projects where applicable
- Mature testing/integration success for defined environments (Akraino and/or partner/upstream projects, which is applicable or both)
- Project artifacts: it is expected that all projects artifacts are available and accessible to all contributors of the Akraino community. Links toward projects artifacts must be provided
- Community
    - Size and diversity of the active community (number and diversity of people contributing)

For each and every review the following steps are required:

- The project review is posted **two weeks** in advance in the Release Wiki. This allows all contributors to provide feedback prior to the review meeting. (include link when available)

- The project review is emailed to akraino-tsc@lists.akraino.org mailing list
- Disposition by TSC: Confirm that the project state is complete and the listed requirements are met.
- Simple majority approval by voting TSC members

Reviews for multiple projects can occur at the same time.

During Release Kick-Off, the project team may request that the TSC conduct a review to move up the ladder.

3.3.7 Project Reviews

3.3.7.1 Incubation Review

The goal of the Incubation Review is to officially launch the project and to support its needs until project Termination Review.

Once a project has passes the Incubation Review, the project is in Incubation State and may span over multiple releases.

Proposal template is available at Proposal Template.

The following artifacts are expected:

- Name of the project is appropriate (no trademark issues etc.); Proposed repository name is all lower-case without any special characters
- Project contact name, company and email are defined and documented
- Description of the project goal and its purpose are defined
- Scope and project plan are well defined
- Resources committed and available
- Contributors identified
- Initial list of committer identified (elected/proposed by initial contributors)
- Meets Akraino TSC Policies
- Proposal has been socialized with potentially interested or affected projects and/or parties
- Cross Project Dependencies (XPDs). In the case where a project will require changes in other projects, those projects are listed in the proposal, and a sponsoring developer in the project has been identified
- Tools have been identified and discussed with relevant partners (Linux Foundation, IT). Once the project pass the review, the tools chain must created within one week. Tools encompass Configuration Management, CI/CD, Code Review, Testing, Team Wiki, End Users documentation (not exhaustive)

3.3.7.2 Maturity Review

The goal of the Maturity Review is to ensure:

- Artifacts for Incubation State are complete and accepted
- Plan for Maturity State are accepted

Once a project has passed the Maturity Review, the project is in Mature State and may span over multiple releases.

Review metrics for Maturity review:

- Successful participation in releases: The project demonstrates stable output (code base, documents) within its history of releases in accordance with the release policy.
- Architecture has been reviewed by the Architecture Committee
- Project is active and contributes to Akraino: The project demonstrates a stable or increasing number of contributions across recent releases. Contributions are commits which got merged to a repository of an Akraino project or a related upstream project. Commits can for example be patches to update the requirements document of a project, code addition to an Akraino or upstream project repository, new test cases and so forth.
- Mature artifacts produced: The project demonstrates that the artifacts produced by the project are deployable (where applicable) and have been successfully deployed, configured and used by end users (typically, service providers).

3.3.7.3 Core Review

The goal of the Core Review is to ensure:

- Artifacts for Maturity State are complete and accepted
- Plan for Core State are accepted. For the Core Review it is expected to deliver a comprehensive integration plan

Once a project has passed the Core Review, the project is in Core State and may span over multiple releases.
Review metrics for Core review include the metrics for maturity review plus the following:

- Contributor diversity: The project demonstrates that it has a stable core team of contributors/committers which are affiliated to a set of at least **three** different companies. Core team members are those who have been active on the project for more than **two** releases, which means they were reviewing contributions to the project in Akraino Code Review and/or in the review-tool of the target upstream project(s).
- Recognized value through other projects: The project demonstrates that its results are leveraged by other Akraino projects in an ongoing way, i.e. for at least the last two releases.
- Successful integration tests (only applicable to projects which provide features/functionality): The project demonstrates that component tests and system-level tests have been implemented,

that tests are used within the Akraino-O CI/CD test pipeline, and that tests bear successful results.

- Stability, Security, Scalability and Performance levels have reached a high bar.

3.3.7.4 Termination Review

The goal of the Termination Review is to ensure that:

- Artifacts for Core state are complete and accepted
- Core project artifacts are acceptable and meet the acceptance criteria
- Project Team has the confidence that its artifacts can be used outside the Akraino community
- Metrics for Termination review are available

3.3.8 Mature Release Process

A Project's Committers make all decisions about Releases of that Project. However, to be eligible to be considered 'Mature', the project must demonstrate a history of following the Mature Release Process. The purpose of the Mature Release Process is to insure openness and maximum opportunity for participation. The idea is to have a simple, clear, public declaration of what a project intends to do and when, and what was actually done in a release cycle. Towards that end, a project following the 'Mature Release Process' should have a Release Plan published at the beginning of its release cycle by its committers, and a Release Review just prior to the project release.

Both Release Plan and Release Review documents are intended to be relatively short, simple, and posted publicly on the wiki to assist project in coordinating amount themselves and the general world in gaining visibility.

These should contain roughly the following sections:

3.3.8.1 Release Plan

- Introduction
- Release Deliverables
- Release Milestones
- Expected Dependencies on Other Projects
- Compatibility with Previous Release
- Themes and Priorities
- Features delivered
- Non-Code Aspects (user docs, examples, tutorials, articles)
- Architectural Issues (if any)
- Security Issues (if any)
- Quality Assurance (test coverage, etc)

- End-of-life (API/Features EOLed in Release)
- Summary of Outstanding Bugs
- Summary of Standards Compliance
- Delta between planned schedule and actual schedule

3.3.8.2 Release Review

- Features delivered
- Non-Code Aspects (user docs, examples, tutorials, articles)
- Architectural Issues (if any)
- Security Issues (if any)
- Quality Assurance (test coverage, etc)
- End-of-life (API/Features EOLed in Release)
- Summary of Outstanding Bugs
- Summary of Standards Compliance
- Delta between planned schedule and actual schedule

3.4 Amendments to the Technical Community Document

The TSC may make amendments to this Technical Community Document at any time with a simple majority of the member of a quorum of the TSC as defined in section 4.4.1.

4 Technical Steering Committee

4.1 Akraino Community Active Contributors

Active Contributors are the cornerstone of the TSC.

Anyone Akraino community member with twenty (20) or more measurable contributions as assessed by the TSC during the previous 12-month period, inclusive of code merged, code reviews performed, wiki page edits, or JIRA activities.

At the transition time the qualification period will be from the point of project inception to the transition date.

4.2 TSC Members

TSC Members consist of an elected subset of up to 20 people from the community's Active Contributors.

Only TSC members have voting rights for decisions taken by the TSC and each TSC member has a single vote.

TSC Membership is bestowed to a subset of Active Contributors through election(s) by the Active Contributors as defined in section 4.4.3.

The TSC functional roles of chair and co-chair will be held by TSC members.

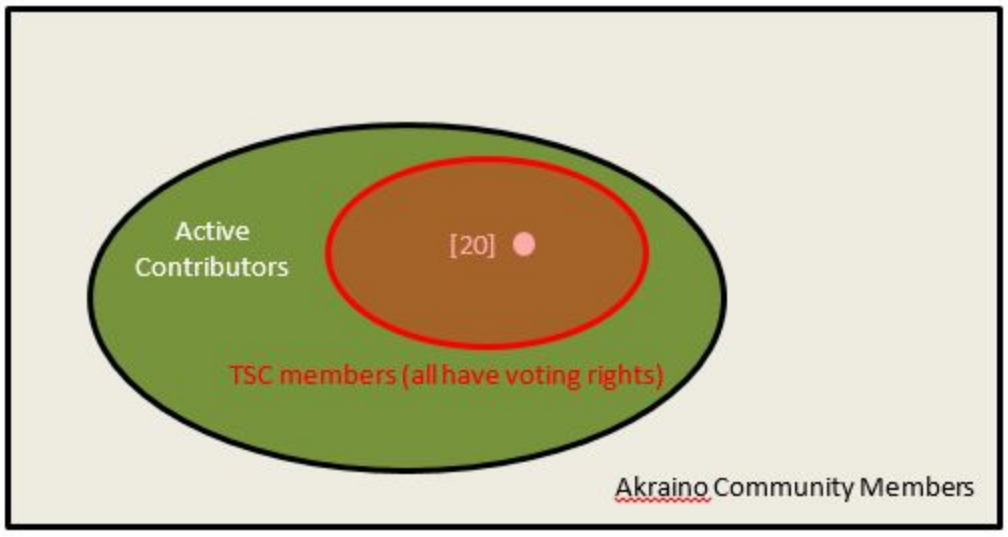

Figure 10 - Akraino Community Structure

4.3 TSC Functional Roles

In addition to Active Contributors and TSC Members there are other roles within the community such as TSC Chair, TSC Co-Chair, Coordinators, project specific committers and others.

Community members with functional roles are not TSC members in their own right.

4.3.1 TSC Chair and Co-Chair

The TSC Chair and Co-Chair are elected by the TSC members.

The TSC Chair and Co-Chair are elected for a term of one year.

The TSC members shall hold elections to select a TSC Chair and Co-Chair at the Transition Point and subsequently every year within one month of the yearly anniversary of the date of the Transition Point.

There are no limits on the number of terms a TSC Chair and Co-Chair may serve.

The Chair and Co-Chair work together to share the responsibilities of the chairing the TSC.

4.3.1.1 Responsibilities

The primary responsibility of the TSC Chair and Co-Chair are to represent the technical community in communications with the LF Networking Fund of The Linux Foundation and to be responsible for:

- Leading TSC meetings;
- This responsibility may be delegated to another TSC Member (in such case, this is to be informed via the TSC email list)
- Representing the technical community to external organizations.
- These responsibilities may be delegated to another member of the technical community.
- Lead the TSC in the execution of the TSCs responsibilities (section 4.3).

4.3.3 Coordinators

4.3.3.1 Coordinator Description

Coordinators are not members of the TSC per se. However a Coordinator may also be a member of the TSC in their own right as an elected Active Contributor.

The TSC has multiple coordinator roles. Each coordinator role comes with its own set of responsibilities to discharge in serving the community via coordinating among various parties.

A coordinator is an **internal** role, so while a coordinator may coordinate among various external liaisons in some instances, **being** a coordinator does not imply being the liaison to any particular external organization or group of organizations.

Coordinators support the TSC Chair in the execution of TSC responsibilities (Section 4.5) and the delivery of Akraino releases. They are responsible for fostering collaboration among the many parties that need to work together to identify, characterize, and solve problems, they do **not** direct solutions.

4.3.3.2 Coordinator origin.

The TSC will solicit nominations for the role. Nominees should have subject matter experience in the relevant coordination area. In the event that multiple candidates self-nominate, the TSC will hold an election as defined in section 4.4.3.3.

The coordinator will regularly report status and issues to the TSC via the Akraino-TSC email list.

4.3.3.3 Coordinator and coordination area lifecycle

There is a lifecycle for the coordinator responsibility (coordination area) and the coordinator appointment.

Coordination Area Creation

A coordination area is created by sending a request to the TSC via the Akraino-TSC email list. The email shall have:

- Email Subject: Creation Request for coordination area: <coordination area name>
- Coordination Area responsibility description: <Description of the coordination area responsibilities>
- Reporting Cadence: <description of when reporting is expected to be delivered to the TSC>
- Area Coordinator: <Name of the TSC nominated area coordinator> (Can be blank)

The decision to create the coordination area is created by a TSC decision by simple majority as defined in section 4.4.1 . A decision can be made with modification, with the modifications captured in the TSC minutes. Once a coordination area is created, it will be documented in the Akraino Wiki.

Coordination Area Update

- A coordination area can be updated by sending a request to the TSC via the Akraino-TSC email list. The email shall have the following: Email subject: Update Request for coordination area: <coordination area name>
- Proposed Update: <Clearly described update of the coordination area. This could be the Area Responsibility Description; Reporting Cadence; Area Coordinator>.

The decision to update the coordination area is created by a TSC decision (according to the TSC voting rules). An update decision can be made with modification, with the modifications captured in the TSC minutes. Once a coordination area is updated, the updates will be documented in the Akraino Wiki.

Coordination Area Termination.

A coordination area can be terminated by sending an email to the TSC via the Akraino-TSC email list. The email shall have the following.

- Email Subject: Close Request for coordination area: <coordination area name>
- Motivation: <Motivation for closing the coordination area>.

The decision to close the coordination area is created by a TSC decision (according to the TSC voting rules). Once a coordination area is updated, the coordination area will be removed from the Akraino Wiki.

4.4 TSC Operations

4.4.1 TSC Decision Making Process

At the transition point and until changed by the TSC there shall be up to 20 seats on the TSC including the chair and co-chair.

Decisions of the TSC should be made by majority vote of TSC Members.

A TSC vote requires a quorum of TSC members to be valid, a quorum consisting of at least 51% of TSC Members present or casting a vote.

Any vote of TSC members that result in a tie shall be considered as not approved.

4.4.2 TSC Election Candidate and Voter Eligibility

4.4.2.1 TSC Members

There are no limitations on the number of candidates that can run in an election for TSC membership, nor is there a limit to the number of candidates from any organization including its subsidiaries (hereon termed simply an 'organization') that can run for TSC membership. However the limits as defined in the TSC technical charter for the total number of TSC members from a given organization shall be enforced by means of the election and interim election process described in **4.4.3**

Candidates must self nominate.

4.4.2.1.1 Candidate and Voter Eligibility

Any Active Contributor community member (regardless of LFN membership), is eligible to run for TSC membership

Any Active Contributor community member (regardless of LFN membership), is eligible to vote in a TSC Member election

Eligibility is effective as of the date and time the nomination process starts

4.4.2.2 TSC Chair and Co-Chair functional roles

Elections for the role of TSC Chair and Co-Chair shall be held immediately after each year TSC Member election.

Candidates must self nominate.

4.4.2.2.1 Candidate and Voter Eligibility

All TSC members are eligible to run for the roles of Chair or Co-Chair.

There are no limits on the number of terms a TSC Chair and Co-Chair may serve.

All TSC Members are eligible to vote in a TSC Chair and Co-Chair election.

4.4.2.3 TSC Coordinators functional roles

The TSC Coordinators shall be elected separately. There is no prohibition against a person holding the role of Chair or Co-Chair and Coordinator or other non TSC role.

Elections are held annually to re-elect existing Coordinator roles. Elections for new Coordination roles are as required. The TSC Members shall elect coordinators from community members as defined in section 4.4.3.3.

Candidates must self nominate (even if nominated by the TSC).

4.4.2.3.1 Candidate and Voter Eligibility

Any community member (regardless of TSC membership) is eligible to serve as a coordinator. Nominees should have subject matter experience in the relevant coordination area.

There are no limits to the number of terms an individual can serve.

4.4.3 TSC Election Mechanics

4.4.3.1 TSC Member Elections

All TSC members shall be elected at the Transition Point and subsequently every year within one month of the yearly anniversary of the date of the Transition Time.

4.2.3.2 TSC Member Election Mechanics

The annual election of TSC Members shall consist of a single stack ranked vote of all candidates to select the remaining 20 TSC members via CIVS https://civs.cs.cornell.edu/

All TSC Active Contributors may cast a single vote

4.4.3.2.1 Enforcement of organization TSC member limits.

The CIVS election wil rank all standing candidates from 1 to N.

If any organization entered more than the permitted limit of electable candidates all excess candidates shall be removed from the results from the least ranked upwards until the organization limit is reached.

Once this is done the remaining 20 highest ranked candidates shall be elected to the TSC.

4.4.3.2.2 Interim elections

In case a TSC member, including Chair or Co-Chair, steps down or is required to step down due to organization limits being exceeded or as a result of company acquisitions or mergers after each yearly election, an interim election may be called by the TSC.

In an interim election an organization may only enter candidates if their current representation on the TSC is below their organization's TSC Member limit.

Interim elections shall otherwise follow all the same procedures and use the same voting schemes as the yearly elections.

4.4.3.3 TSC Chair/Co-Chair/Coordinator Election Mechanics

The TSC will elect from amongst TSC Members a chairperson and co-chairperson for a term of one year.

The TSC shall hold elections to select a TSC Chair and Co-Chair at the Transition Point and subsequently every year within one month of the yearly anniversary of the date of the Transition Point.

Election of a TSC Chair/Vice Chair/Coordinator shall use a multiple-candidate method shall consist of a single stack ranked vote of all candidates using CIVS https://civs.cs.cornell.edu/

4.5 Responsibilities of the TSC.

Subject to the Technical Charter, the TSC is responsible for:

- Defining Akraino's release vehicles (such as a Coordinated Release) that align with the Project's,mission,
- Fostering cross-project collaboration,
- Serving as Akraino's primary technical liaison body with other consortiums and groups,
- developing an architecture,
- setting simultaneous release dates,
- defining release quality standards,
- defining technical best practices and community norms (including the establishment and maintenance of a Development Process),
- monitoring technical progress,
- mediating technical conflicts between Committers and PTLs,
- organizing inter-project collaboration,
- coordinating technical community engagement with the end-user community.

4.6 TSC Subcommittees

The TSC, at its discretion, may establish subcommittees to assist the TSC with its responsibilities and provide expert guidance in technical subject areas (e.g., architecture or security).

The TSC may amend or introduce alternative approaches to convene and run subcommittees.

4.6.1 Membership

4.6.1.1 Subcommittee Membership Eligibility

It is expected that subcommittee membership shall be open to all Akraino Contributors; however, subcommittees [or TSC] may impose restrictions such as the number of participants from a single company [organization]. While the desire may be to keep its size and scope limited, each subcommittee shall be open to the Akraino membership.

4.6.1.2 Subcommittee Chair / Vice Chair

Each subcommittee may elect a Chair and optionally a Co-Chair who is responsible for leading meetings and representing the subcommittee to the TSC.

4.6.1.3 Subcommittee Chair / Co-Chair Elections

The Chair or Co-Chair will be elected by members of the subcommittee as of the date the nomination process starts for the election.

4.6.1.4 Subcommittee Voter Eligibility

All members of the subcommittee are eligible to vote.

4.6.1.5 Subcommittee Election Confirmation

The elected Chair (and/or Co-Chair) is submitted to the TSC for confirmation. The TSC decides to accept the outcome or requests a new voting.

4.6.2 Advisory role

Subcommittees are advisory in nature, and not authoritative. They provide advice to projects and to the TSC.

Subcommittees operate on a rough consensus basis. If the subcommittee is unable to reach consensus on what advice to offer, the subcommittee Chair shall raise the issue with the TSC, where a formal vote can be taken, or advise the project that the subcommittee cannot reach consensus.

4.6.3 TSC subcommittee lifecycle.

4.6.3.1 Creation of a TSC subcommittee

The TSC decides the creation of a subcommittee in accordance with TSC decision procedure.

In order to create a TSC subcommittee, a TSC member shall make a proposal to the TSC (via Akraino-TSC email list) that that shall cover at least the following:

- TSC subcommittee name.
- TSC subcommittee purpose: <Description of subcommittee purpose>
- TSC subcommittee expected deliverables:<List of expected deliverables>

- TSC subcommittee starting participants: <List of TSC nominated participants>
- Optionally TSC subcommittee definition of done: <Description of what is expected at the conclusion of the subcommittee. This may relate to the deliverables>

In addition to the TSC nominated participants, members of the community who wish to participate shall self nominate to the TSC and shall become Subcommittee members without election

Members of the sub-committee who want to run for Chair for the subcommittee self nominate

4.6.3.2 Update of a TSC subcommittee

The TSC can modify a TSC subcommittee via a TSC decision. To request such a modification, a request is made to the Akraino-TSC email list.

4.6.3.3 Conclusion of a TSC subcommittee

The TSC decides the termination of the TSC subcommittee in accordance with the TSC decision procedure. The submission of a request to terminate the TSC subcommittee should cover:

- TSC subcommittee name
- TSC subcommittee deliveries: <description of what has been achieved>
- Motivation for termination of TSC subcommittee: <Reason for requesting the termination of the subcommittee>

4.6.4 Subcommittee vs. coordinator

As a guideline, a subcommittee is most appropriate when the task to be addressed involves a relatively stable group of people with a high level of intersection of common involvement. A coordinator is more appropriate when there is a more dynamic group of people and issues may change frequently. A coordinator is also more appropriate for smaller efforts or topics requiring infrequent meetings.

Glossary

Term	**Full Meaning**
IoT	Internet of things
PTL	Project technical lead
ETE	End to end
SDK	Software development toolkit
API	Application program interface
POD	Point of delivery

CI/CD	Continuous integration and continuous delivery
VNF	Virtual network function
LCM	Lifecycle management
YAML	YAML Ain't Markup Language. It's basically a human-readable structured data format.
OS	Operating system
EOL	End of life
VNF	Virtual Network Function (VM or container based)
CIVS	Condorcet Internet Voting Service