Thanks to all community & TSC members provided inputs and feedback

This slide is the consolidation of all such inputs.

Draft – Yet to be baselined by TSC
Scope of this slide/discussion

• Intent of this slides is not to repeat or cover the content of “TSC charter document”.
• The content articulated on this slide is focused on the additional details that TSC need to baseline as “TSC community document”.
• This presentation is to kick start the discussion, followed by content documented in the wiki.
• TSC appreciates the feedback shared by the community and this presentation incorporated such feedbacks.
Technical mission

1. Create end to end configuration for a particular Edge Use case which is complete, tested and production deployable { Blueprints}.

2. Develop projects to support such end to end configuration. Leverage upstream community work as much as possible to avoid duplication. { Projects}.

3. Work with broader edge communities to standardize edge apis { Socialization, so this community tools can interoperate}.

4. Encourage Vendors and other communities to validate VNFs & edge application on top of Akraino blueprints { Facilitate a eco-system}.

AKRAINODEGGE STACK
Akraino Project summary

1. For Simplicity call everything as a project
2. Support three types of projects – Feature projects, Integration, validation
3. Feature Projects
   1. Primary goal is to liaison with upstream project to fill in gaps in the upstream code needed by the edge blueprint(s)
   2. Or develop projects with in community which are not supported in the upstream
   3. Do not fork upstream projects [upstream first]
   4. Project focus area for this community – Common user experience across blueprints, Edge Testing, Integration/Ops/security tools
   5. Primary upstream community – based on what is used within the blueprints.
4. Integration projects
   1. Blueprints are integration projects which integrates multiple components for a edge Point-of-Delivery (POD)
   2. Blueprints define the fundamental characteristics / components of any Point-of-Delivery (POD) instantiation
   3. Blueprints should be complete, tested and production deployable
   4. Maintain the Continuous integration at the Akraino Community
   5. Leverage Vendor & Community labs to demonstrate the Continuous deployment and feed back the results to the community to ensure working of “a blueprint”
5. Vendor & Community labs
   5. Akraino community to establish guidelines to connect with Akraino CI and CD feedback to LF.
Akraino Project Types & Scope

Akraino – Project(s)

Upstream
- Code Development – Features Projects
  - Common user experience
    - Akraino Portal + workflows
  - Edge Testing
    - Blueprint ETE
    - Testing suites
    - CI/CD scripts
  - Integration & ops & Security tools
    - xxxx

Within Akraino
- Integration – Projects
  - Seed
    - Network cloud – Telco use case – OpenStack/ONAP/K8/Docker/Airship/OS agnostic based
  - Pipeline
    - ONF SEBA
    - Real-time RAN
    - IOT
    - Etc.,

Validation (Apps/VNFs) – Projects
- Akraino Validation Labs (Future)

Vendor/other community Labs
- Infrastructure s/w
  - ONAP
  - OpenStack
  - K8
  - Docker
  - OS
- Integration tools
  - Airship
  - Starlingx
  - Etc.,
- Collaborate standards
  - APIs (EdgeX Foundry,...)
- Legend
  - Continuous Integration (CI)
  - Continuous Deployment (CD)

9/20/18
# How to arrive at the blueprints? – 5 step process

<table>
<thead>
<tr>
<th>Sequence</th>
<th>What</th>
<th>Definition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Edge Use Case</td>
<td>Description of the business outcome. Defines workload characteristics, design constraints, cost range, etc.</td>
<td>Community member to use &quot;template&quot; and submit for TSC review</td>
</tr>
<tr>
<td>2</td>
<td>Edge Use Case Specification</td>
<td>Specifications (HW/SW components, deployment configurations, etc.) designed to support Use Case(s) and described in a testable, implementation-agnostic manner (&quot;what&quot;, not &quot;how&quot;).</td>
<td>Community member to use &quot;template&quot; and submit for TSC review</td>
</tr>
</tbody>
</table>
| 3        | Blueprint | - Reference Architecture to meet the use case need.  
- Implementation-specific declarative configuration file(s) ready to be consumed by that implementation’s deployment and LCM tool(s) and resulting in a stack that passes the design’s tests. |  
- Developed and maintained within the Akraino Community (CI)  
- Project team maintained |
| 4        | Validation | - Tested without VNF/Edge Apps – prove it works  
- Tested with VNF/Edge Apps – Prove ETE works |  
- Akraino community process  
- Tested in Vendors, Providers, Community labs  
- Results published under the blueprint |
| 5        | User Deployment | Production deployment by users/providers/vendors |  
- Provide feedback to the community (bug and enhancement reports) |

Thanks to Frank for providing inputs
### Akraino Use Cases and Use Case Specifications

#### Akraino Use Cases Templates

- Business driven

#### Use Case Characteristics

<table>
<thead>
<tr>
<th>Use Case Characteristics</th>
<th>Network Cloud Use Case Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Need</td>
<td>Network based edge cloud that can be deployed at provider data center and telco offices</td>
</tr>
<tr>
<td>User Experience</td>
<td>Single Pane of Glass control - Administrative and User Based GUIs</td>
</tr>
<tr>
<td>Cost Of Solution</td>
<td>Less 800K a POD [46 servers deployment] - Cruzer POD configuration</td>
</tr>
<tr>
<td>Scale</td>
<td>Minimum 10 – Maximum 1000 Locations</td>
</tr>
<tr>
<td>Applications</td>
<td>Any type of Edge Virtual Network Functions</td>
</tr>
<tr>
<td>Power restrictions</td>
<td>Less than 50K watts</td>
</tr>
</tbody>
</table>

#### Akraino Use Case Specifications

Specifications (HW/SW components, deployment configurations, etc.) designed support Use Case(s) and described in a testable, implementation-agnostic manner

<table>
<thead>
<tr>
<th>Use Case Specifications</th>
<th>vEPC service on Network Cloud Specification Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Workload</strong></td>
<td>vEPC or any Edge VNFs</td>
</tr>
<tr>
<td><strong>Infrastructure orchestration</strong></td>
<td>OpenStack/ONAP</td>
</tr>
<tr>
<td><strong>UCP tool</strong></td>
<td>Airship</td>
</tr>
<tr>
<td><strong>Workload Characteristics</strong></td>
<td>VMs and Containers</td>
</tr>
<tr>
<td><strong>Under cloud</strong></td>
<td>K8 &amp; Docker</td>
</tr>
<tr>
<td><strong>SDN</strong></td>
<td>SR-IOV &amp; OVS-DPDK</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>Linux (Ubuntu)</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td>X86 based G10 and above servers.</td>
</tr>
</tbody>
</table>

Sample templates – not a final version

The Linux Foundation Internal Use Only
Existing vs. new blueprints

- Categorize blueprints by Family { e.g., Network Cloud}
- "A" blueprint can support multiple POD types { e.g., Cruzer ( 6 racks) , Unicycle ( 1 rack), Rover (single server) }
- "A" Pod could support multiple “configuration types” but within the criteria defined reference architecture for that blueprint { e.g., different Linux OS}
- "A" configuration type is a defined by declarative file { e.g., YAML for the POD type}
- Each committer/project submitter should look at existing blueprint and see if it can support their use case by existing configuration or with new configuration type
- If existing blueprint does not support the use case or with new configuration type then to submit a new blueprint proposal to TSC
- TSC to review the blueprint proposal and approve/disapprove
- Intention is to maximize the "configuration types” supported by a blueprint and minimize the number of blueprint. Discretion applied during review process.
Relationship Between Blueprint Specs & PODs

Blueprint Specifications define the declarative configuration for each deployment model or Point of Delivery (POD) of a Blueprint.

- YAML files allow for different configurations within the same blueprint

<table>
<thead>
<tr>
<th>Blueprint Level</th>
<th>POD Specification Level</th>
<th>Component Level</th>
<th>Declarative Configuration Level - YAML File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family: Network Cloud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unicycle</td>
<td>Ubuntu/OS/ODL based</td>
<td>{yaml files U1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Centos/OS/ODL based</td>
<td>{yaml files U2}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ubuntu/OS/Neutron based</td>
<td>{yaml files U3}</td>
</tr>
<tr>
<td></td>
<td>Tricycle</td>
<td>X</td>
<td>{yaml files T1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
<td>{yaml files T2}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Z</td>
<td>{yaml files T3}</td>
</tr>
<tr>
<td></td>
<td>Rover</td>
<td>A</td>
<td>{yaml files R1}</td>
</tr>
</tbody>
</table>

- **Point of Delivery** (POD) - 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.

Choices shown is just for illustration and not recommendations.
Thanks to Andrew for providing inputs.
Analogy from Andrew’s slide

Family: Canidae
- Genus: Canis (wolf like genus)
  - Species: Familiaris (domesticated dog)
  - Species: Lupus (grey wolf)
  - Species: Rufus (red wolf)
- Genus: Vulpes velox (fox like genus)
  - Species: Vulpus (red fox)
  - Species: Lagopus (arctic fox)
  - Species: Zerda (fenec fox)
- Genus: Speothos velox (South American genus)
  - Species: Venaticus (Bush dog)

Family: Felidae ……
Blueprint Components vs. specifications

- Blueprint Specifications (a.k.a. Declarative configurations) are built from the component options for the layers contained within a Blueprint.
- Blueprint Specifications can evolve in subsequent releases to add / remove functional layers
- Declarative configuration naturally allows ways to support different components within a same blueprint

<table>
<thead>
<tr>
<th>Blueprint Components</th>
<th>UCP tool</th>
<th>SDN</th>
<th>Overcloud</th>
<th>Undercloud CNI</th>
<th>Undercloud</th>
<th>Host OS layer</th>
<th>HW layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Airship</td>
<td>SR-IOV/OVS</td>
<td>OS Ocata</td>
<td>Calico</td>
<td>KBs 1.9</td>
<td>Ubuntu 14.04</td>
<td>Dell R720/ HP DL360</td>
</tr>
<tr>
<td></td>
<td>Airship</td>
<td>ODL Boron</td>
<td>OS Pike</td>
<td>Multus</td>
<td>KBs 1.12</td>
<td>Ubuntu 16.04</td>
<td>HP DL360</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TitaniumFabric R1</td>
<td>k8s</td>
<td>Flannel</td>
<td></td>
<td>Centos 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TitaniumFabric R2</td>
<td></td>
<td></td>
<td></td>
<td>Centos 7</td>
<td></td>
</tr>
</tbody>
</table>

Selections show one possible specification within this blueprint

This is for illustration and doesn’t contain all layers required for the NC blueprint

Thanks to Andrew for providing inputs
“Feature projects” relationship to “Integration projects”

- Feature project could be specific to a blueprint or across the blueprint
- Integration project = a blueprint
- A Feature Project is a long term endeavor setup to deliver features across multiple releases, which have a shorter lifespan
- A Integration project is a long term endeavor setup to deliver ETE functionalities across multiple releases
- Integration project requires at least one continuous deployment lab supported by vendor or a community. Without such CD lab, blueprint working cannot be validated.
### Project lifecycle – States and Reviews

- To move from one state to the next state, the Project Team must obtain TSC approval based on a set of evaluation criteria.
- Project teams request TSC reviews to move up the ladder. TSC majority approval is required to advance from one state to the next.
- Same process for Feature and Integration projects.

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Incubation</th>
<th>Mature</th>
<th>Core</th>
<th>Archived</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project doesn’t exist yet</td>
<td>Project has resources</td>
<td>Project is fully functioning and stable</td>
<td>Project provides value to and receives interest from a broad audience.</td>
<td>Project can reach Archived state for multiple reasons</td>
</tr>
<tr>
<td>May not have real resources</td>
<td>Project is in the early stages of development</td>
<td>Project has achieved successful releases</td>
<td>Not expected to be used in production environments.</td>
<td>Project has successfully completed and artifacts provide business values, or project has been cancelled for unforeseen reasons</td>
</tr>
<tr>
<td>Proposal to be create project due to business needs.</td>
<td>The outcome is a minimum viable product (MVP) that demonstrates the value of the project and is used to collect feedback</td>
<td></td>
<td></td>
<td>Project in any state can be Archived through a Termination Review.</td>
</tr>
</tbody>
</table>
Release plan

- Akraino releases will include a set of project deliverables.
- Akraino releases can be composed of 1 to N projects.
- Akraino projects are long term endeavors setup to deliver features across multiple releases, which have a shorter lifespan.

The scope of each project is aligned with the Akraino charter and the scope of each release is defined with the objective to fulfill a particular EDGE use case(s).
TSC Subcommittees

- The TSC may establish subcommittees to assist the TSC with its responsibilities and provide expert guidance in technical subject areas.

  - Subcommittees are advisory in nature, and not authoritative. They provide advice to projects and to the TSC.
  - Subcommittee Members - Each subcommittee shall determine its own membership eligibility, in consultation with the TSC.
    - Subcommittee Chair - Each subcommittee may elect a Chair and optionally a Vice-Chair who is responsible for leading meetings and representing the subcommittee to the TSC.
Next Steps

› Baseline this deck to agree on the terminology for the community
› Review this slide with the community (September 20th) to get community feedback
› Work on the content of Akraino Technical community document – Early draft available in the google drive
› Target to baseline the document by end of September or early October.
Additional backup slides
**What is Akraino Blueprint?**

Blueprints – Approved and tested declarative configuration based on use cases, set of Hardware & Software, Point of delivery (POD).

Reference Architecture – Defines Akraino building blocks

Declarative Configuration – Hides lower layer complexity to user

CI/CD, Integration & Testing Tools – Drive product quality

Akraino release – End Product

---

**Blueprints**

1. **Blueprints1**
2. **Blueprints n**

**Reference Architecture**

For Edge Use cases

---

**Declarative Configuration**

- "A" Hardware & Software
- "N" Hardware & Software

**POD**

**CI/CD, Integration, & Testing**

**TSC will provide acceptance criteria for release**
Why Akraino Blueprint?

Benefits:
- Low Cost
- Large Scale
- Zero Touch Provisioning
- Industry Adoption
- Life Cycle Support
- Proven and Tested by Community
- Fully Integrated End to End Solution (CI / CD)
- Production Quality
Blueprints with clear business need

**Use Cases**

- **Network Cloud**
  - Single to Multiple racks
  - Telecom (5G, ...)

- **Real Time**
  - Single – Dual Server/White boxes
  - Access (RAN), PON, IOT, ...

- **Customer-premises / Far Edge**
  - All-in-one White boxes
  - Universal CPE
**Network Cloud Blueprint (Seed Code)**

**AT&T Network Cloud Blueprint**

- **Use Case Based**
  - Telco / 5G / Enterprise Use Cases

- **Fully Integrated ETE Solution (CI / CD)**
  - Airship based
  - Upstream Integrated
  - Full CI in LF
  - Automated CD Validation Using Real Hardware

- **Proven and Tested by Community**
  - Community Developed and Maintained

- **Life Cycle Support**
  - Continuous Integration
  - Documentation

- **Production Quality**
  - Production deployed at AT&T
For More Information, Please Visit www.akraino.org
Proposals from Community members – incorporated in the above deck.
Backup materials
Akraino Blueprints and Blueprint Specification/Templates

A framework proposal V4.0
Family: Canidae

Genus: Canis (wolf-like genus)
  - Species: Familiaris (domesticated dog)
    - Species: Lupus (grey wolf)
    - Species: Rufus (red wolf)

Genus: Vulpes velox (fox-like genus)
  - Species: Vulpus (red fox)
    - Species: Lagopus (arctic fox)
    - Species: Zerda (fenec fox)

Genus: Speothos velox (South American genus)
  - Species: Venaticus (Bush dog)

Family: Felidae
Choices shown are just for illustration and not recommendations
Blueprints and Blueprint Specification/Release Templates

At the highest level the Blueprint defines the fundamental must have characteristics/components of any POD deployed using it e.g. A “Network Cloud” Blueprint deploys OpenStack using a k8s undercloud with Airship based LCM (etc)

These are immutable attributes - if they are omitted or replaced a different Blueprint results

Can be considered an Akraino POD’s Family

Within a given blueprint a POD’s deployed components can be tailored by different Blueprint Specifications e.g. At each Akraino release of the Network Cloud blueprint its Blueprint Specification Template would contain the set of all verified possible plugins/options for each layer

Can be considered an Akraino POD’s Genus

The exact POD configuration of a given Blueprint Specification is the last level of description e.g. This is the contents of the yaml manifests for a Network Cloud blueprint’s POD

Can be considered the final definitive definition of deployment. An Akraino POD’s Species

Validation of hosted applications (e.g. VNFs) against a Blueprint and its Specification is then possible

---

### Network Cloud Blueprint Specification Template Release 1

<table>
<thead>
<tr>
<th>SDN</th>
<th>None (neutron)</th>
<th>ODL Boron</th>
<th>TitaniumFabric R1</th>
<th>TitaniumFabric R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overcloud</td>
<td>OS Ocata</td>
<td>OS Pike</td>
<td>k8s</td>
<td></td>
</tr>
<tr>
<td>Undercloud CNI</td>
<td>Calico</td>
<td>Multus</td>
<td>Flannel</td>
<td></td>
</tr>
<tr>
<td>Undercloud</td>
<td>K8s 1.9</td>
<td>K8s 1.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host OS layer</td>
<td>Ubuntu 14.04</td>
<td>Ubuntu 16.04</td>
<td>Centos 6</td>
<td>Centos 7</td>
</tr>
<tr>
<td>HW layer</td>
<td>Dell R720</td>
<td>HP DL360</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is for illustration and doesn’t contain all layers required for the NC blueprint

Red box selections show one possible specification within this blueprint
Network Cloud Blueprint and Specification/Release Templates

Different Blueprints would have different options to select in the Blueprint Specification as the functionality deployed in such a POD would be different.

   e.g. an IOT blueprint may not use OpenStack as a virtualization

The Specification Template of a given Blueprint can evolve in subsequent releases to add / remove functional layers.
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description of the business outcome / use case to be achieved, incl. workload characteristics, design constraints, etc.</td>
<td>Edge Stack + workloads (VNFs, edge apps, ...) that together solve the described business use case.</td>
</tr>
<tr>
<td>Example: Network Cloud</td>
<td>Example: A vEPC service hosted on the Network Cloud</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edge Stack Design (Specification + Tests)</th>
<th>Edge Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification of an edge stack (HW/OS components, deployment config, etc.) designed to address a given (group of) Use Case(s) and described in a testable, implementation-agnostic manner (&quot;what&quot;, not &quot;how&quot;).</td>
<td>An edge stack deployment that meets that stack’s design specification and passes the corresponding tests.</td>
</tr>
<tr>
<td>Example: Single-rack stack with a Kubernetes cluster for infra services (Ceph, ONAP, ...), an OpenStack cluster for NFV tenant services, HA-configuration, configured with network segregation, ...</td>
<td>Example: A deployed Kubernetes and OpenStack cluster with running ONAP, EdgeX, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blueprint</th>
<th>Blueprint LCM Tool(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation-specific (set of) declarative configuration file(s) ready to be consumed by that implementation's deployment and LCM tool(s) and resulting in a stack that passes the design's tests. Example: Airship site design configuration files.</td>
<td>Tool that deploys and operates an edge stack according to the Blueprint and artifacts (images, secrets, ...) it receives from the Arkano CI/CD system. Example: Airship.</td>
</tr>
</tbody>
</table>

Author – Frank Zdarsky
Several orgs work on specifying this interface. The User edge should say what interfaces it supports.

How the User edge node interact with the Provider edge, directly or through Data center only, registration, security, data reporting etc... A neutral interface is needed.

The User edge can interact with the Data center system directly. This is the base case interface. A neutral interface is needed.

How the User edge node interact with the Provider edge, directly or through Data center only, registration, security, data reporting etc... A neutral interface is needed.

Out of scope

**Devices**

A.k.a. Sensors, actuators, camera, ...

We may need devices for use case testing, but in general they should be considered out of scope for Akriano.

**User edge**

A.k.a. IoT gateway, enterprise, CPE ...

If a blueprint covers user edge, a ‘software stack’ for the “User edge” should be provided.

**Provider edge**

A.k.a. Network edge, Infrastructure edge, telco edge ...

If a blueprint covers Provider edge, a ‘software stack’ for the “Provider edge” should be provided.

**Regional data center**

**Central data center**

A.k.a. Cloud ...

This should be seen as an abstract entity for Akriano. Its internals should be out of scope. It can public cloud or private cloud or telco cloud etc.