
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

Building and Managing k8s with k8s

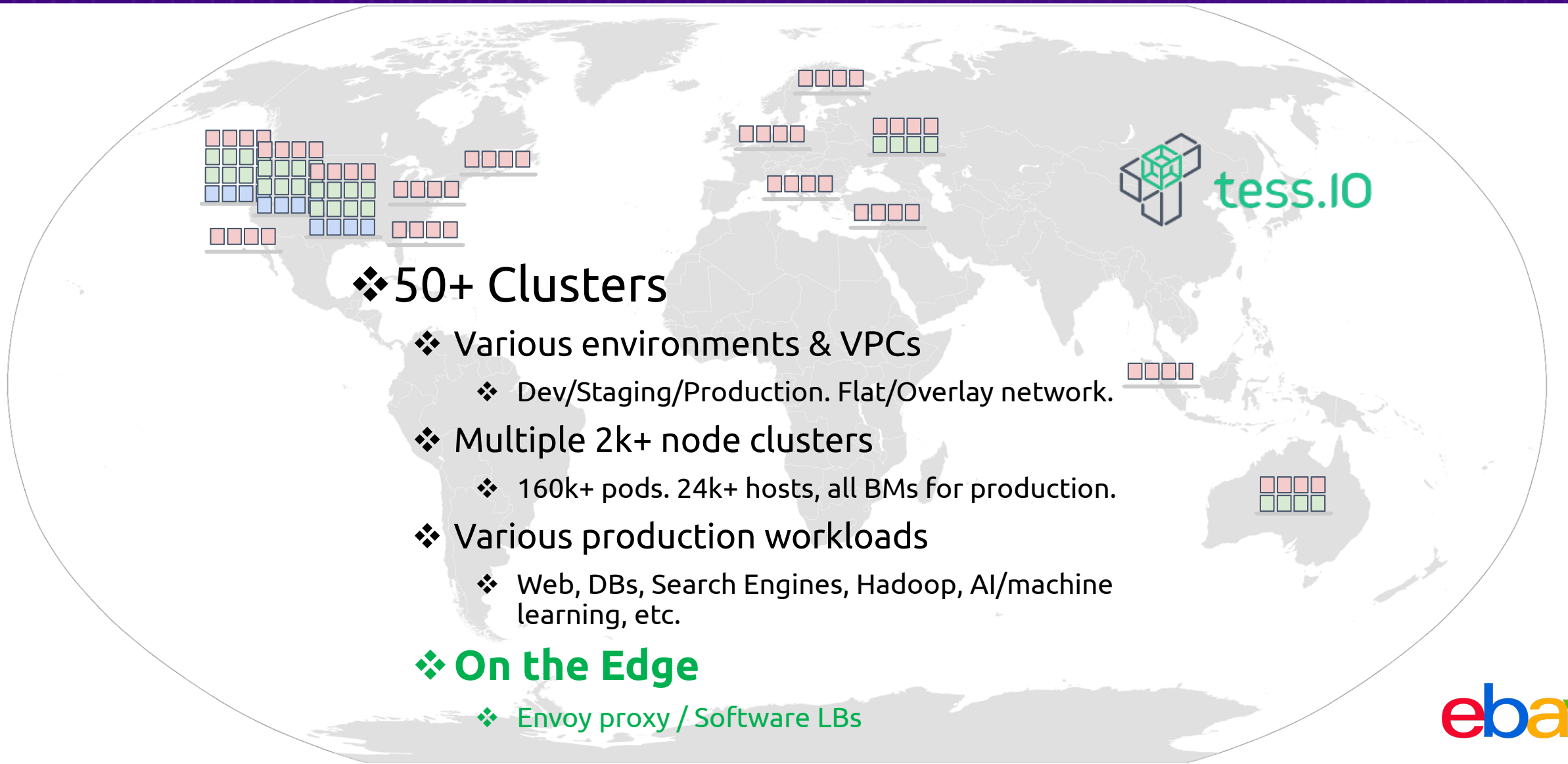
eBay's fleet management system and GitOps practices

Xin Ma *xima@ebay.com*

Agenda

- eBay's k8s deployments
- Fleet management system – Architecture View
- Build & Manage IaaS with k8s
  - BM provisioning system (homegrown)
  - VMs from k8s
  - Other providers, OpenStack, GCP, etc.
- Build & Manage k8s with k8s
  - Salt operators and Deployments
- GitOps
- Features inspired by k8s
- Summary, Q&A

eBay's k8s deployments



- 50+ Clusters
  - Various environments & VPCs
    - Dev/Staging/Production. Flat/Overlay network.
  - Multiple 2k+ node clusters
    - 160k+ pods. 24k+ hosts, all BMs for production.
  - Various production workloads
    - Web, DBs, Search Engines, Hadoop, AI/machine learning, etc.
  - **On the Edge**
    - Envoy proxy / Software LBs

ebay

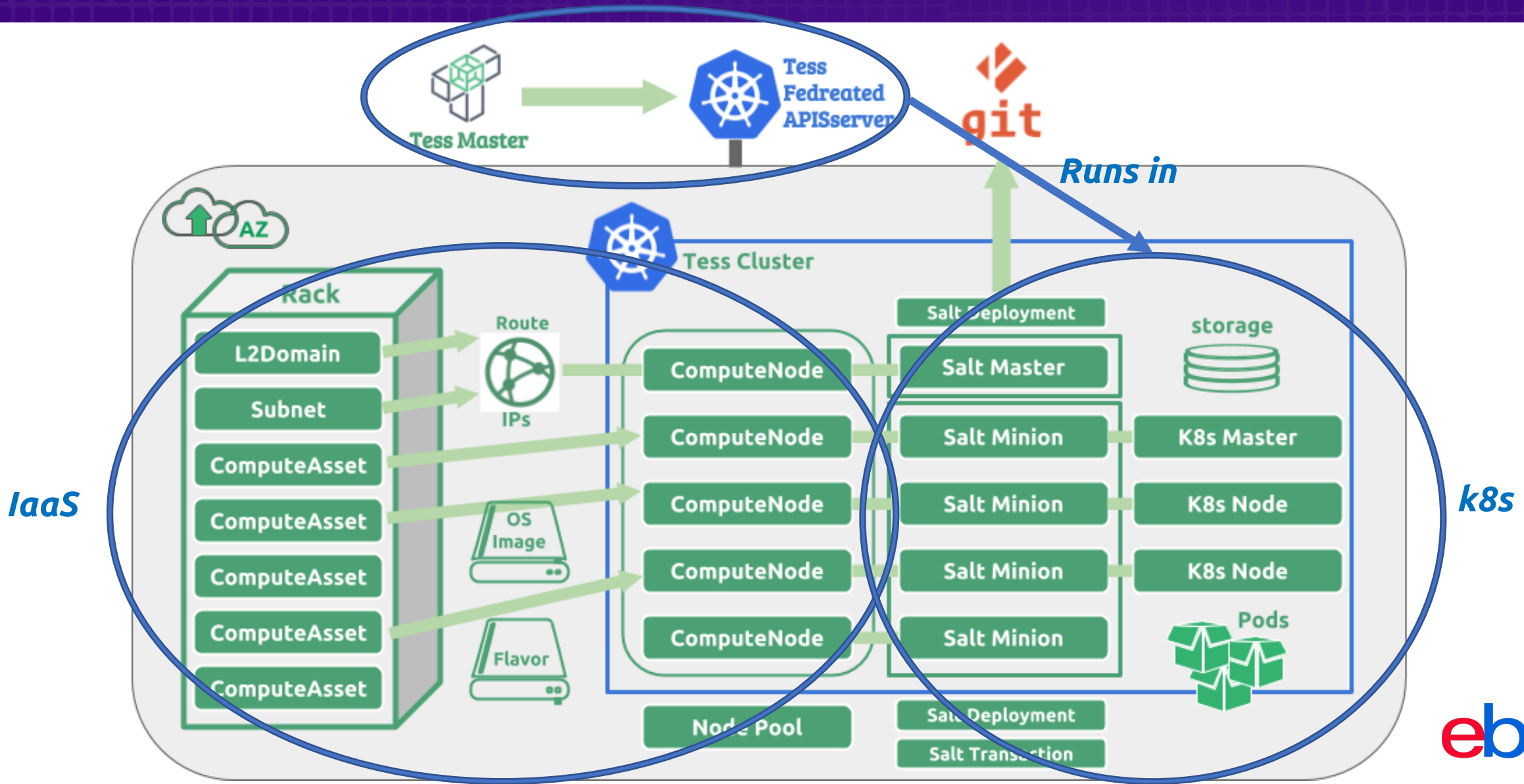
Fleet management system

- Architecture View

Fleet management system

- Fleet management system
  - Runs with an apiserver + etcd-operator at any k8s cluster
  - CRDs and Modeling
    - Hardware (DataCenter).
      - *Racks, Switches, VPC, Subnets, IPaddress, etc.*
    - Software stack.
      - *OSImages, ComputeNodes, SaltMaster & SaltMinions to deploy k8s, etc.*
  - Controllers
    - Provision *ComputeNodes* like pods, create *NodePools* like deployments
      - multiple providers support, and a built-in homegrown BM provisioning system.
    - Install *SaltMaster* on *ComputeNode*
      - From a git commit id of Salt repo
    - Install k8s at computenodes, with *SaltMinions*
      - roles for kube master and nodes.
    - Install & manage cluster with *SaltDeployments*, just like deployments
    - Remediations, transactions, scheduler, and rolling updates, etc.

Architecture view (single AZ)



ebay

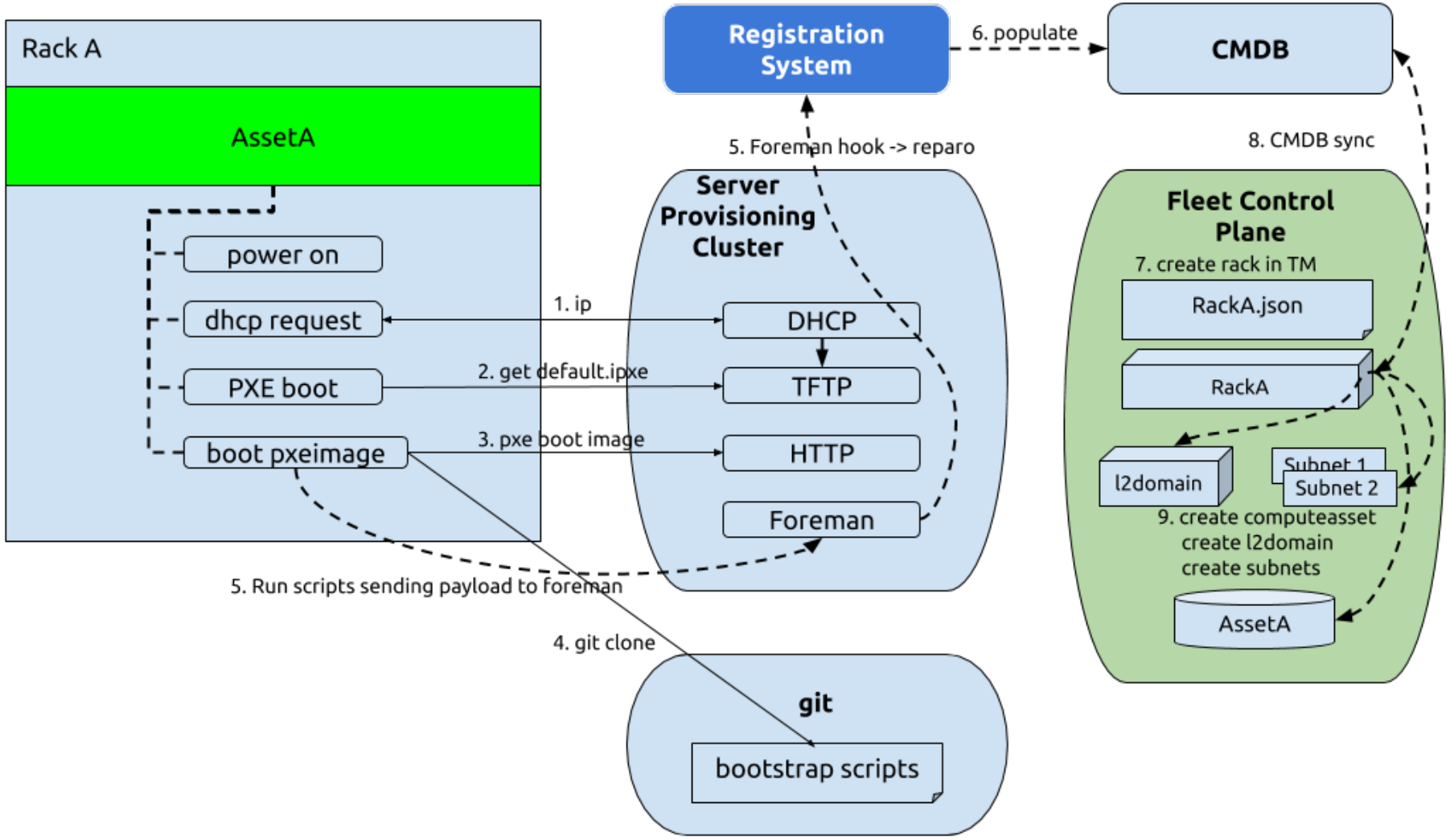
Build & Manage IaaS with k8s

Build & manage IaaS with k8s

- k8s needs computes, from *Providers*
  - private. Openstack
  - public. GCP, azure, etc.
- Define and implement interfaces for providers
  - Controllers (provisioning, remediation, node, k8s/salt controllers, etc.)
  - We support openstack and GCP
- How about building a provider with k8s
  - Homegrown BM provisioning system.
  - Self-contained, and highly scale
- Manage & orchestrate the fleet
  - Models the datacenter
  - Manage the compute lifecycle
- How about providing VMs from k8s
  - Virtlet

Bootstrap and Model the Datacenter

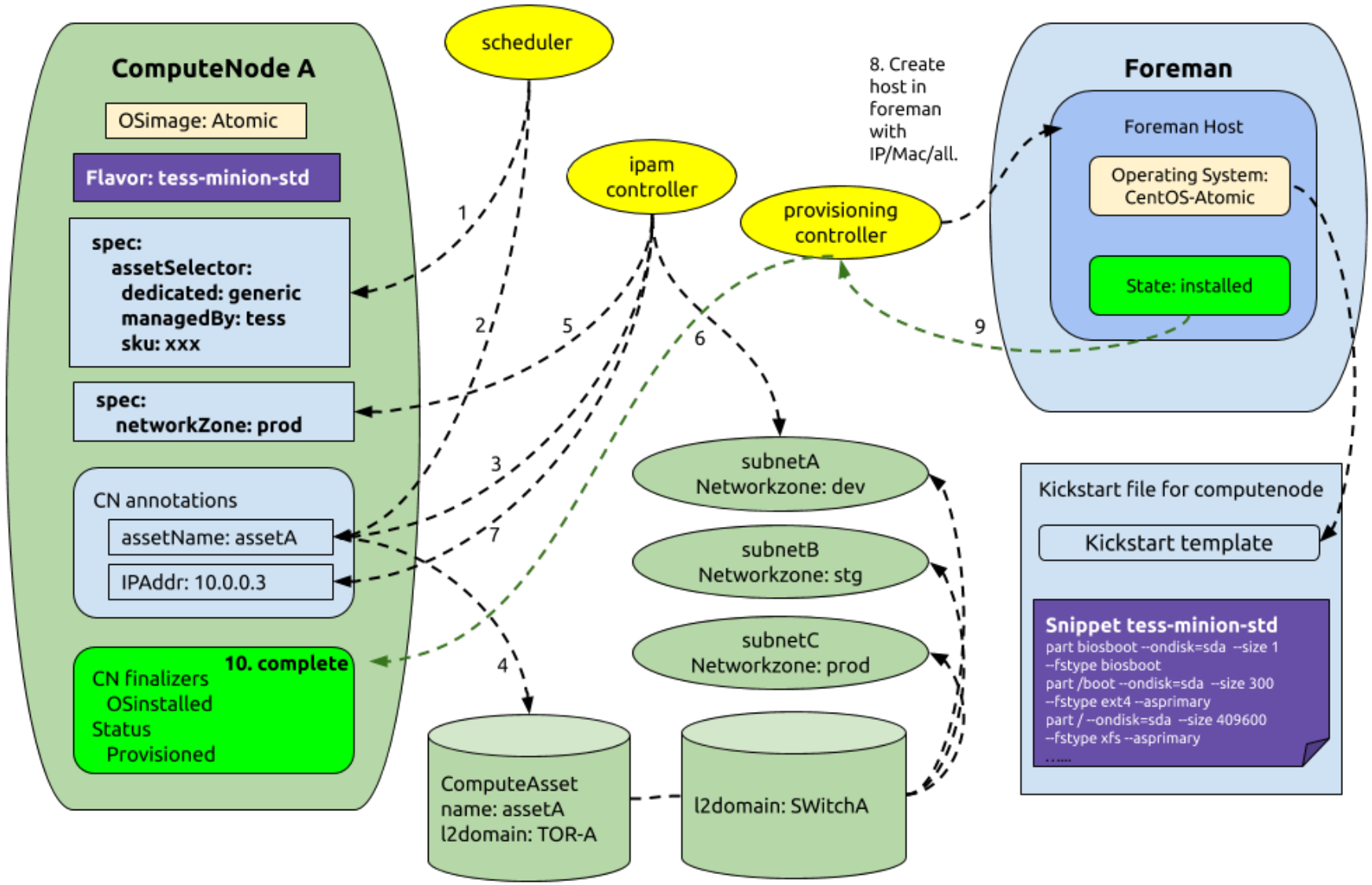
KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019



- Models the Fleet
  - Racks
  - L2domains
  - Subnets
  - Routes
  - ComputeAssets
  - VPCs
    - Networkzone
- Auto create Models
- Lots of CRDs

```
apiVersion: infra.tess.io/v1alpha1
availabilityZone: {}
devices:
- function: network
  index: 1
  l2Domain:
    name: l2domainA
  macAddress: XX:XX:XX:XX:XX:XX
  networkSwitch:
    name: switchA
  speed: 10000Mb/s
  type: Provision
- function: network
  ipAddr: 10.2.2.2
  l2Domain:
    name: l2domainLOM
  macAddress: YY:YY:YY:YY:YY:YY
  networkSwitch:
    name: lomSwitchA
  type: Management
kind: ComputeAsset
manufacturer: dell
metadata:
```

BM Provisioning System



- Provisioning BMs
  - Create computenode
  - Set tess native provider
  - Scheduler picks assets
  - Ipam assigns IP
  - Osimage maps to ks template
  - Flavor creates partition table
  - Create host in foreman
  - Wait for host to install
  - Register in CMDB
  - Create DNS
  - etc.
- Controllers works in order
  - Finalizers
  - Annotations

ebay

VMs on k8s

- VMs on k8s
  - Kata
  - Virtlet / Kubevirt
- Kata: runtime for secure workloads
- Virtlet: a compute provider for VMs
  - VM is created as a Pod in k8s cluster
  - Run k8s on top of k8s
  - Non-container workloads, like windows VMs.

```yaml
apiVersion: compute.tess.io/v1alpha1
kind: NodePool
metadata:
  name: node-pool-virtlet
  namespace: virtlet
spec:
  replicas: 4
  selector:
    node-pool-name: node-pool-virtlet
  template:
    metadata:
      generateName: demo-
      labels:
        node-pool-name: noc
    spec:
      flavor: vm-flavor-tir
      osImage:
        name: ubuntu18.04-s
      provider: virtlet
      sshKeys:
      - name: sshkey-demo
```

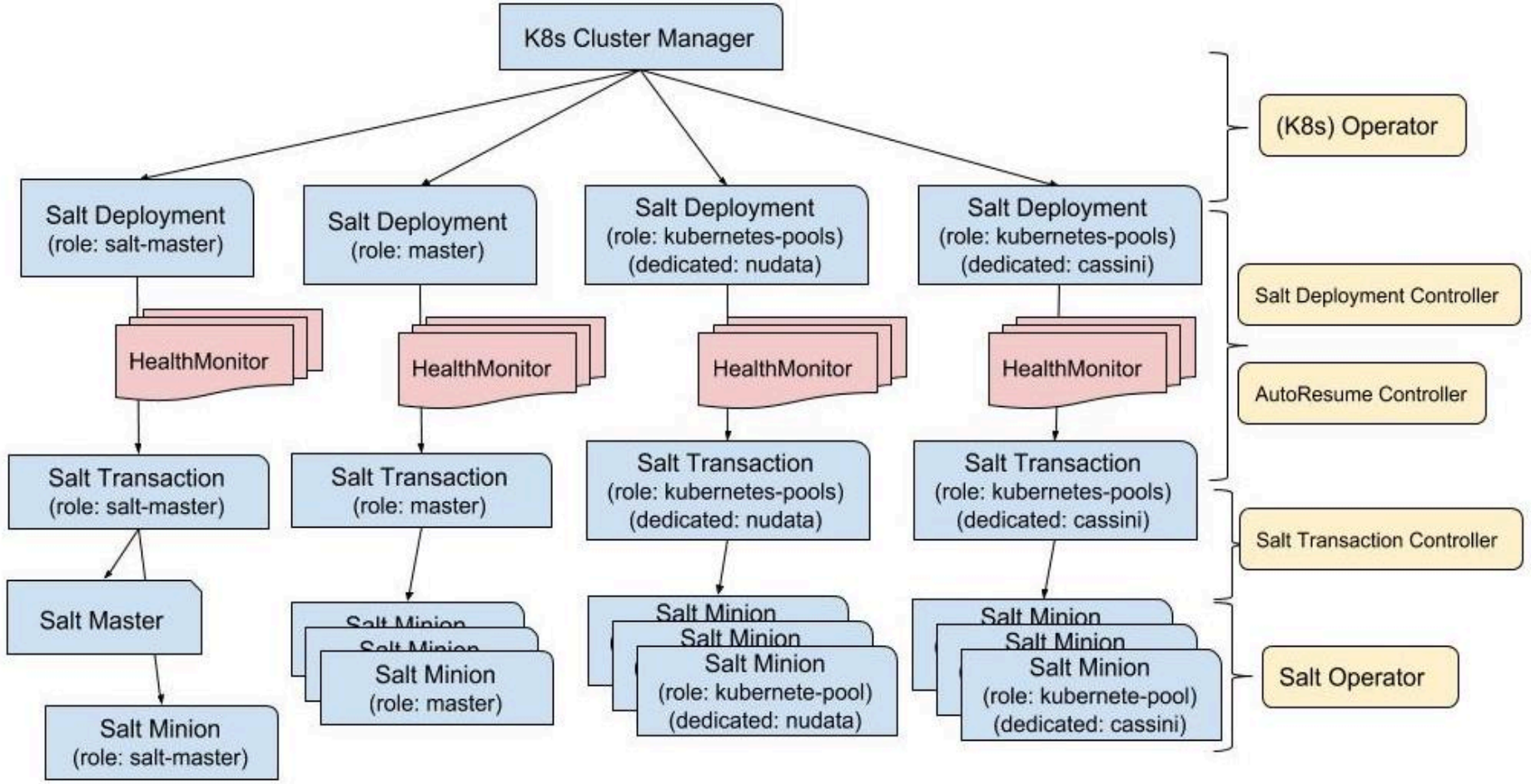
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: demo-vm-1
  annotations:
    VirtletRootVolumeSize: 30Gi
    VirtletNetworkDevice: virtio-net-pci
    VirtletDiskDriver: virtio
    VirtletCPUModel: host-model
    VirtletSSHKeys: |
      ssh-rsa xxxx
    VirtletVCPUCount: "4"
    kubernetes.io/target-runtime: virtlet.cloud
    VirtletCloudInitImageType: configdrive
spec:
  containers:
  - name: linux-vm
    image: virtlet.cloud/ubuntu18.04-server-amd64.qcow2
    imagePullPolicy: IfNotPresent
    resources:
      limits:
        cpu: "4"
        memory: 16Gi
      requests:
        cpu: "4"
        memory: 16Gi
    stdin: true
    tty: true
```

Build & Manage k8s with k8s and GitOps

Build & manage k8s with k8s

- Build k8s with salt.
- Models Saltstack
  - Salt Master
  - Salt Minions
  - Salt Deployment
    - Aligns with, and manage nodepools of one or a group of computenodes.
    - Deploys SaltMaster or SaltMinions
      - Build salt master from a salt state git repo (commit id)
      - Deploy salt minions for k8s master, and minions
  - Salt Transactions
    - Rolling updates.
      - buckets
- Controller: Salt operator

Building & managing k8s with salt operators



- Salt deployment
  - 1-1 maps to *nodepool*
  - Salt master salt deployment
    - Create *saltMaster*
      - At computenode
    - From git repo
      - Commit or branch
  - Salt minion salt deployment
    - Apply kube master states
    - Apply kube node states
- Salt transactions
  - Create buckets
    - Based on Strategy
      - Rack by rack
      - custom
    - *SaltMinions*
  - Rolling updates
- Auto resume controller
  - Bucket by bucket
    - Based on health monitor
  - Automated rollout

GitOps for devops

- **Every bit is in git**
  - OSTree to build OS, Salt to config OS – *Build Operating System*
  - Provisioning Hosts by creating CRDs – *Build Computes*
  - Salt installs and config k8s – *k8s Deployment*
  - Compute and cluster *lifecycle management* with remediation contro*llers*
  - CRDs and controllers do most of the work.
    - check CRDs into git – *Fleet in git*
- GitOps for devops
    - PR driven operations, everything declarative.
- Consistency & Scalability

This is how we scale, and expand our k8s footprints in eBay

Features inspired by k8s

- Scheduler
  - Picking free assets for BM provisioning system
  - Selectors, Affinity and anti-Affinity
- Rolling updates
  - Transactions and buckets
  - Strategies (pluggable)
- Disruption budgets
  - ComputeNode disruption budgets
    - restart / reboot / delete
- Health Monitor framework
- probes, and conditions, for CRDs
  - ComputeNode liveness and readiness probes.
    - Collect probes and conditions from alert manager and node problem detector
  - Build the compute remediation controller


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2019
Unified Fleet Mgmt. with k8s & GitOps
Manage the Fleet with CRDs and controllers
Model and manage the Fleet
BM and VM provisioning based on k8s
Managing k8s with salt operators.
Every bit is in git.
Gitops for devops
Consistency & Scalability
Q&A
ebay
tess.IO