MIRANTIS

MCP

April 2017

From Installer-Centric to Operations-Centric Distro

Installer-Centric Distro

Pros

- Easy to install
- Great to learn & gain initial experience

Cons

- Operations (post-install actions) is an afterthought. Difficult to change, update, upgrade
- Upgrades are disruptive, happen once every 12+ months
- Monolithic architecture, hard to customize deployments in the field

Operations-Centric Distro

Pros

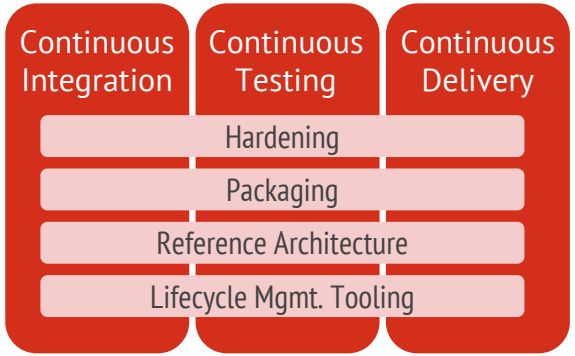
- Unified tooling for initial deployment and ongoing management of the cloud
- Drive and audit changes through version control system
- Update and upgrade capabilities are built into the platform (Operating System, Kubernetes, OpenStack, SDN, Ceph, etc)
- Component-based architecture, deployment flexibility

Cons

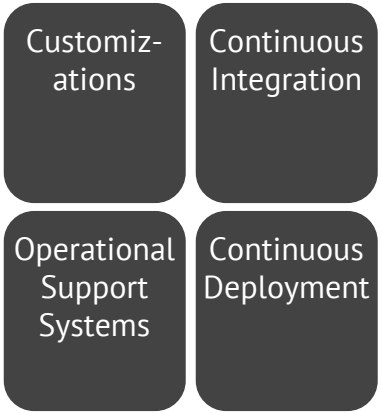
- Slightly more complicated initial deployment

MCP: Continuously Delivered Private Cloud

LEADING OPEN-SOURCE COMMUNITIES



MIRANTIS ENGINEERING



CUSTOMER

Continuous delivery for both applications AND infrastructure

Key Architecture Concepts

MCP is built around the following architecture concepts

Configuration Store

- Configuration is versioned and has a central configuration repository
- Allows to update/rollback a cloud to a known state. And audit configuration at any point in time

CI/CD

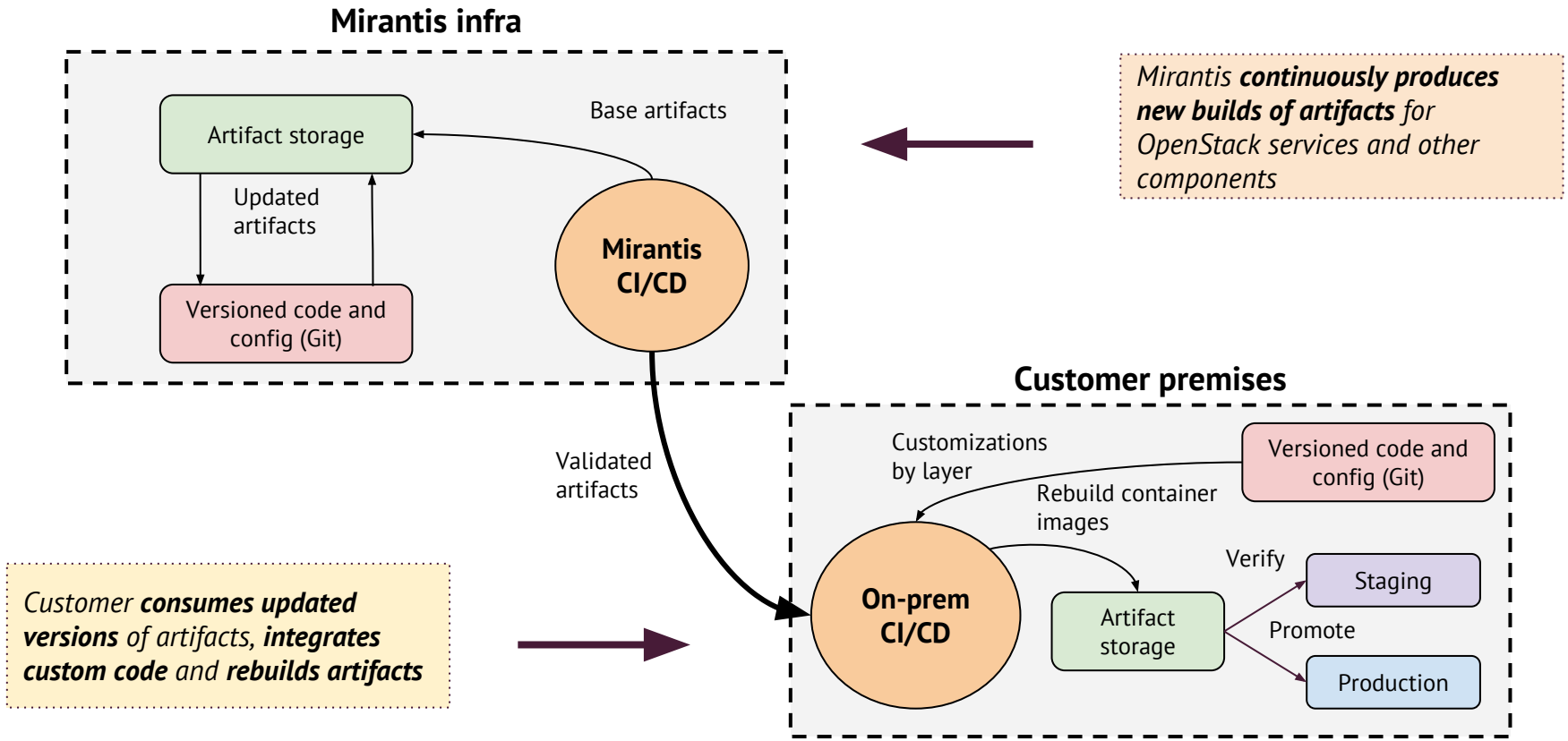
Jenkins

- Run Infrastructure as code
- All changes (configuration changes, code changes, new components) get pushed through CI/CD pipeline

Containers

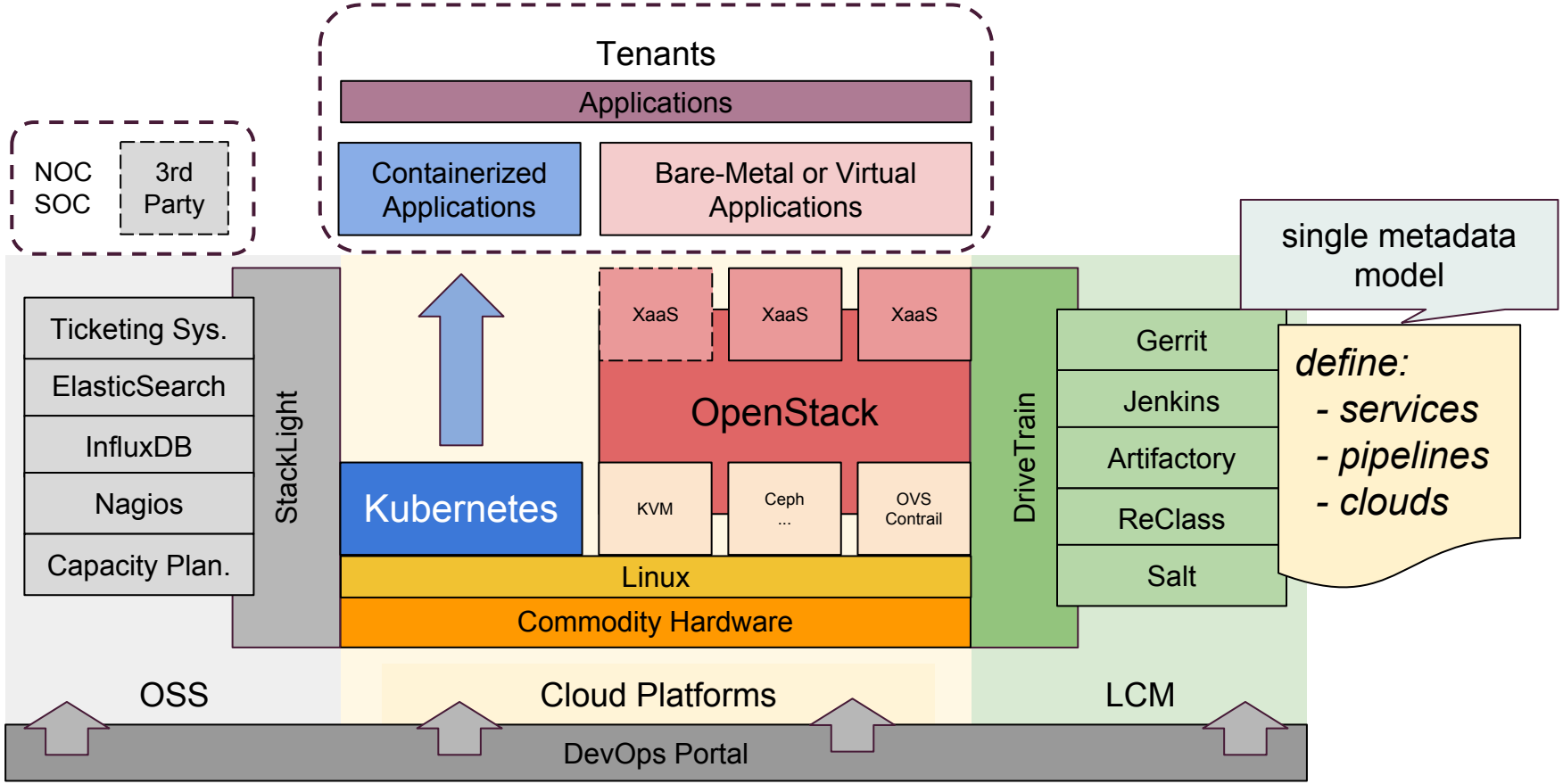
- State of the system is primarily defined by the binary footprint of containers
- Allows atomic upgrade/rollback operations on per-component basis

Artifact Delivery Pipeline



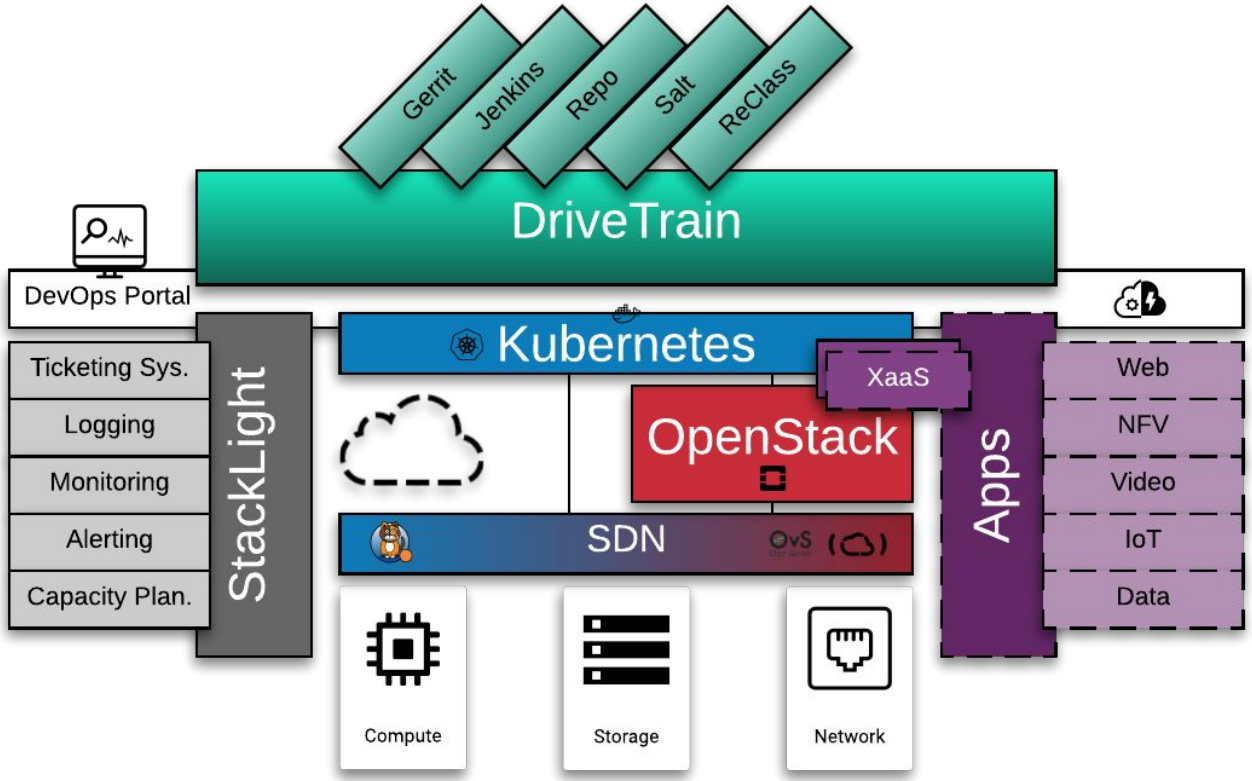
What it looks like

Flexible, modular and scalable architecture (reference implementation for IaaS + CaaS)



What it looks like

Flexible, modular and scalable architecture (reference implementation for IaaS + CaaS)



MCP 1.0: Mandatory v optional components

	Component	Mandatory?
LCM	Salt	yes
	Reclass	yes
	Jenkins	yes
	Gerrit	yes
Cluster Software	Kubernetes	no
	OpenStack	no
	OpenContrail	no
Cluster Analytics	StackLight: Logging Monitoring and Alerting	no
	DevOps Portal	no

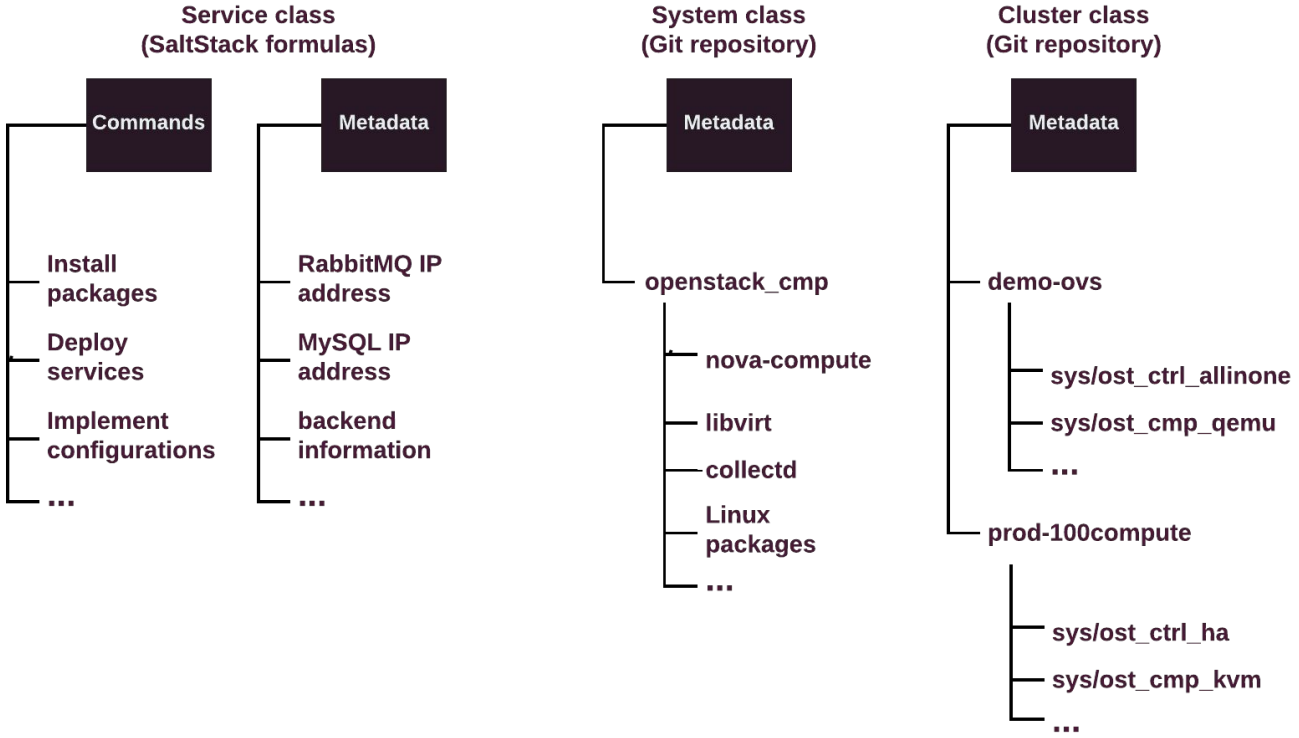
Delivered as Artifacts:

- Linux / Containers
- OVS, QemuKVM etc
- OpenStack
- OpenContrail
- K8S
- Calico
- Salt Formulas
  - Service Metadata
  - Execution Logic
- OSS Tooling (StackLight components, DevOps Portal)

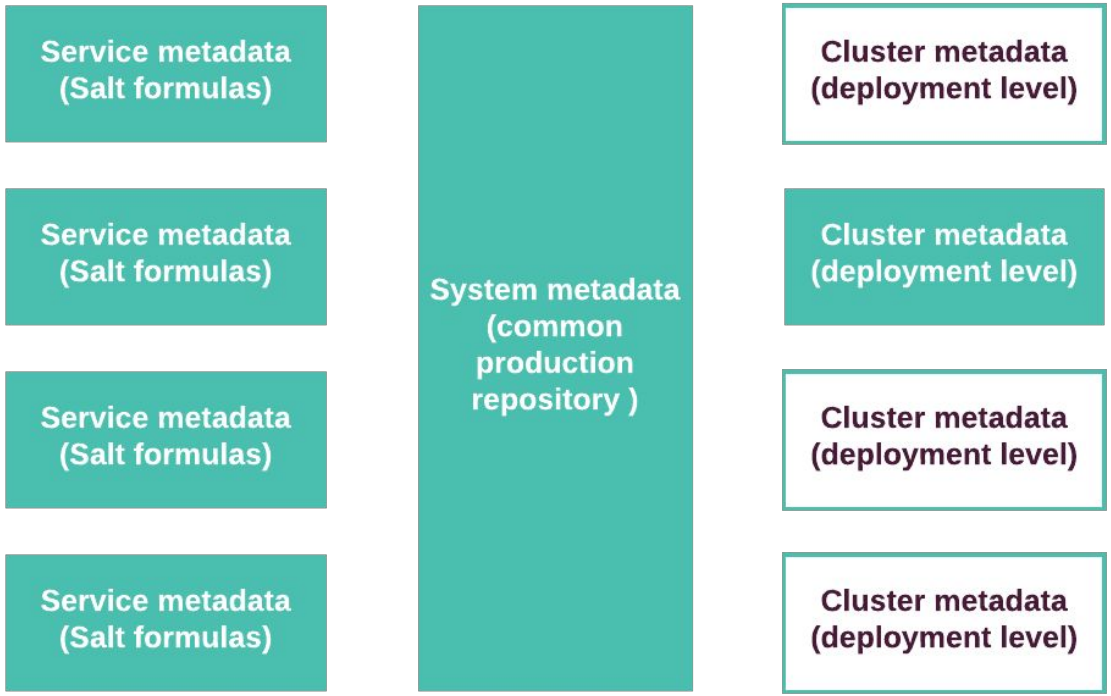
Delivered as uncompiled code (via GIT):

- Metadata (state definition)
  - System (integration metadata)
  - Cluster (implementation specific metadata)
- Jenkins Pipelines
  - Shared Libraries (Groovy)

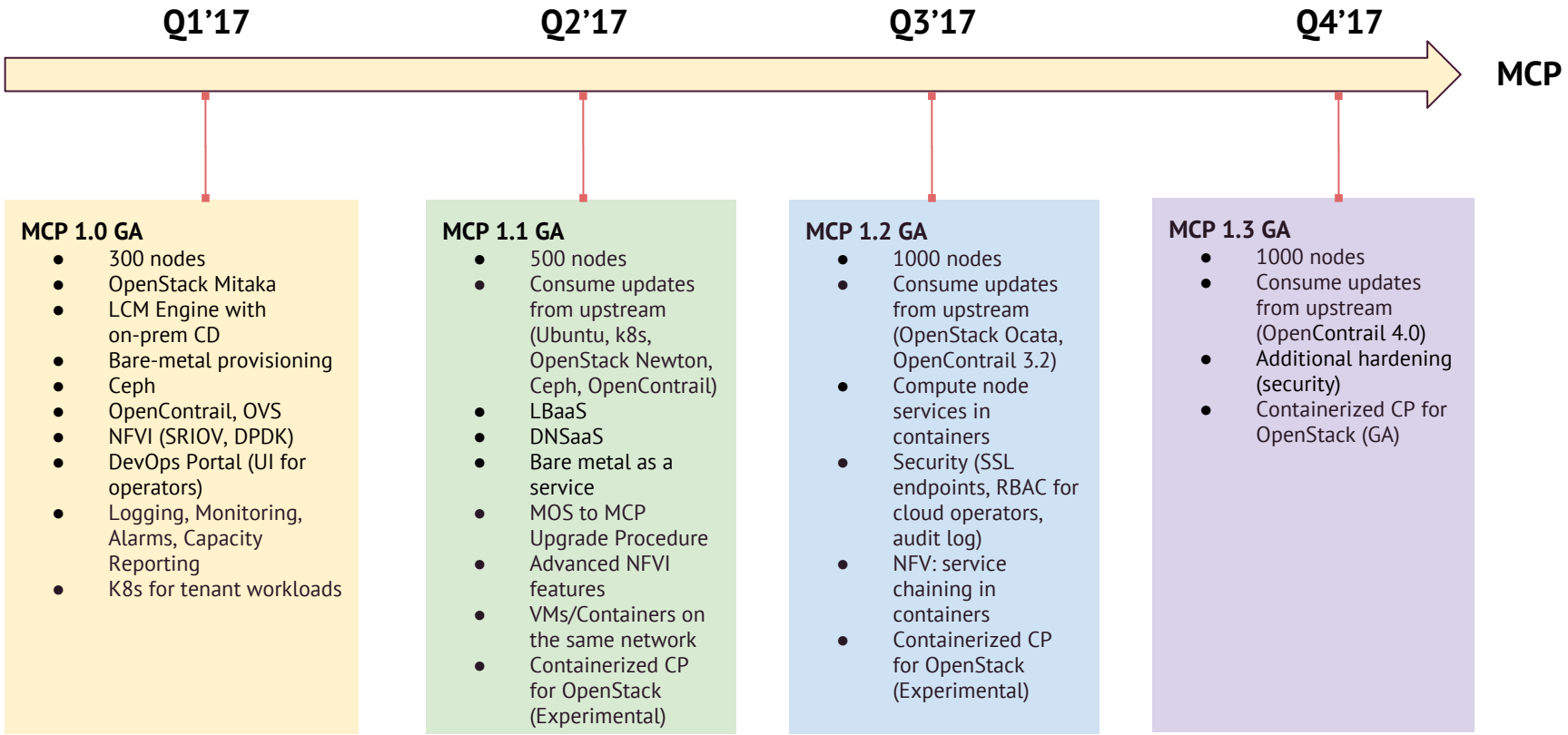
Anatomy of the Metadata Model



Metadata Model Repository Structure



MCP (2017) - High-Level Roadmap



** tentative schedule/scope*