Introduction to pulp-operator

Mike DePaulo
mikedep333@redhat.com
@mikedep333 [GitHub, Freenode]

Talk Overview

- Bottom-Up (concepts and our usage)
    - Pulp Containers
    - Pulp Kubernetes manifests
    - Pulp Operator
- Related tooling:
    - plugin-template.git (CI)
    - pulp-insta-demo.sh
    - pulp-demo.git
- Future development

Pulp Containers: pulpcore.git/containers

docker

podman

Why Containers instead of VMs?

- Performance advantages
  - lxc model
- Cloud VMs changed paradigm
  - Separate data from code
- Application packaging & distribution
  - The real reason
  - Eliminating fights between devs and ops
  - Images vs containers

Proper container design

- Single process/service
- Microservices made possible
- Increases need for orchestration layer (Kubernetes)

Dockerfiles

- Like an RPM spec
- Horribly inflexible syntax
- Our Dockerfile is now jinja2-templated

Jinja2-Rendered Dockerfile (abridged) (1 of 3)

```
FROM fedora:30

RUN echo 'LANG="en_US.UTF-8"' > /etc/locale.conf

ENV LANG=en_US.UTF-8
ENV PYTHONUNBUFFERED=0
ENV DJANGO_SETTINGS_MODULE=pulpcore.app.settings
ENV PULP_SETTINGS=/etc/pulp/settings.py

RUN           dnf -y update && \
              dnf -y install wget git && \
              dnf -y install libxcrypt-compat && \
              dnf -y install python3-psycopg2 && \
              dnf -y install glibc-langpack-en && \
              dnf -y install python3-createrepo_c && \
              dnf -y install libmodulemd-devel && \
              dnf -y install python3-libmodulemd && \
              dnf -y install libcomps-devel && \
              dnf -y install python3-libcomps && \
              dnf clean all

RUN ln -s /usr/bin/python3 /usr/bin/python
RUN ln -s /usr/bin/pip3 /usr/local/bin/pip
```

Jinja2-Rendered Dockerfile (abridged) (2 of 3)

```
RUN mkdir -p /etc/pulp

RUN pip install gunicorn

RUN pip install pulpcore
RUN pip install pulpcore[postgres]

RUN pip install pulpcore-plugin

RUN pip install pulp-certguard
RUN pip install pulp-file
RUN pip install pulp-ansible
RUN pip install pulp-cookbook
RUN pip install pulp-docker
RUN pip install pulp-maven
RUN pip install pulp-python
RUN pip install pulp-rpm
```

Jinja2-Rendered Dockerfile (abridged) (3 of 3)

```
COPY pulpcore/containers/images/pulp/container-assets/wait_on_postgres.py /usr/bin/wait_on_postgres.py
COPY pulpcore/containers/images/pulp/container-assets/wait_on_database_migrations.sh /usr/bin/wait_on_database_migrations.sh
COPY pulpcore/containers/images/pulp/container-assets/pulp-common-entrypoint.sh /pulp-common-entrypoint.sh
COPY pulpcore/containers/images/pulp/container-assets/pulp-api /usr/bin/pulp-api
COPY pulpcore/containers/images/pulp/container-assets/pulp-content /usr/bin/pulp-content
COPY pulpcore/containers/images/pulp/container-assets/pulp-resource-manager /usr/bin/pulp-resource-manager
COPY pulpcore/containers/images/pulp/container-assets/pulp-worker /usr/bin/pulp-worker

ENTRYPOINT ["/pulp-common-entrypoint.sh"]
```

Examples of image name & tag?

- registry/repository/image:tag

- quay.io/pulp/pulp:latest
- quay.io/mikedep333/pulpcore:3.0.0rc4
- localhost/pulp:3.0.0-pr123
    - pulp:3.0.0-pr123

- Remember: image vs container

Tooling around the dockerfile

- Example vars.yaml used for template:
  - - pulp_master_plugins_master:
  - image_name: pulp
  - tag: latest
  - pulpcore: git+https://github.com/pulp/pulpcore.git#egg=pulpcore
  - pulpcore_plugin: git+https://github.com/pulp/pulpcore-plugin.git
  - plugins:
  - - "git+https://github.com/pulp/pulp-certguard.git"
  - - "git+https://github.com/pulp/pulp_file.git"
  - - "git+https://github.com/pulp/pulp_ansible.git"
- vars.yaml also accepts:
  - Stable pip install strings like “pulp_file”
  - ./pulp_file (required a lot of work)
- Ansible build.yaml
  - Generates Dockerfile from vars.yaml & Dockerifle.j2
  - Calls `docker build` (or `buildah`)

4 containers in one image?

“RUN” not specified.

4 scripts - 1 for each type of container

- pulp-api
- pulp-content
- pulp-worker
- pulp-resource-manager

What about collectstatic & migrations

- No database available at container build time
- **Currently** done via pulp-api script

Pulp Kubernetes manifests: pulp-operator.git

kubernetes

Why add orchestration on top of container runtimes?

- Define relationships between single service/process-containers
- Multiple container hosts
- Storage and networking not fully fleshed out
- Running ensures daemon desired state of overall application is both reached & maintained

Kubernetes (“K8s”) for Orchestration (1 of 2)

- Container infrastructure
  - Storage
  - Networking
  - Compute
- Objects include:
  - Deployments
  - Containers/Pods
  - Services / Routes
  - (Persistent) Volume Claims
- Understands:
  - Services
  - Their relationships
  - Whether they are up or down

Kubernetes (“K8s”) for Orchestration (2 of 2)

- Uses “namespaces” to isolate apps
- Components:
  - Controller (running daemon / management server)
  - Nodes (managed container hosts)
- Configuration files for defining Kubernetes objects:
  - Declarative
  - Define desired state of the objects
  - Often says “use generic interface”, and lets infra use desired implementation plugin

Kubernetes Distributions

- From most featured to least-featured:
  - Openshift
  - Upstream Kubernetes / minikube
  - K3s (used by pulp-operator CI, plugin-template CI, and pulp-insta-demo.sh)
- Note: There are many more

pulp-api.deployment.yaml (1 of 3)

```
apiVersion: v1
kind: Deployment
metadata:
  name: pulp-api
  namespace: "{{ project_name }}"
  labels:
    app: pulp-api
spec:
  replicas: {{ pulp_api.replicas }}
  selector:
    matchLabels:
      app: pulp-api
  template:
    metadata:
      labels:
        app: pulp-api
```

pulp-api.deployment.yaml (2 of 3)

```
...
spec:
  ...
  template:
    ...
    spec:
      containers:
        - name: pulp-api
          image: "{{ registry }}/{{ project }}/{{ image }}:{{ tag }}"
          imagePullPolicy: "IfNotPresent"
          args: ["pulp-api"]
          env:
            # TODO: Replace with k8s secrets
            - name: PULP_ADMIN_PASSWORD
              value: "password"
          ports: # (related to "service" object)
            - protocol: TCP
              containerPort: 24817
```

pulp-api.deployment.yaml (3 of 3)

```
...
spec:
  ...
  template:
    ...
    spec:
      containers:
        - name: pulp-api
          ...
          volumeMounts:
            - name: pulp-server
              mountPath: "/etc/pulp/"
            - name: pulp-file-storage
              readOnly: false
              mountPath: "/var/lib/pulp"
      volumes:
        - name: pulp-server
          configMap:
            name: pulp-server
            items:
              - path: settings.py
                key: settings.py
        - name: pulp-file-storage
          persistentVolumeClaim:
            claimName: pulp-file-storage
```

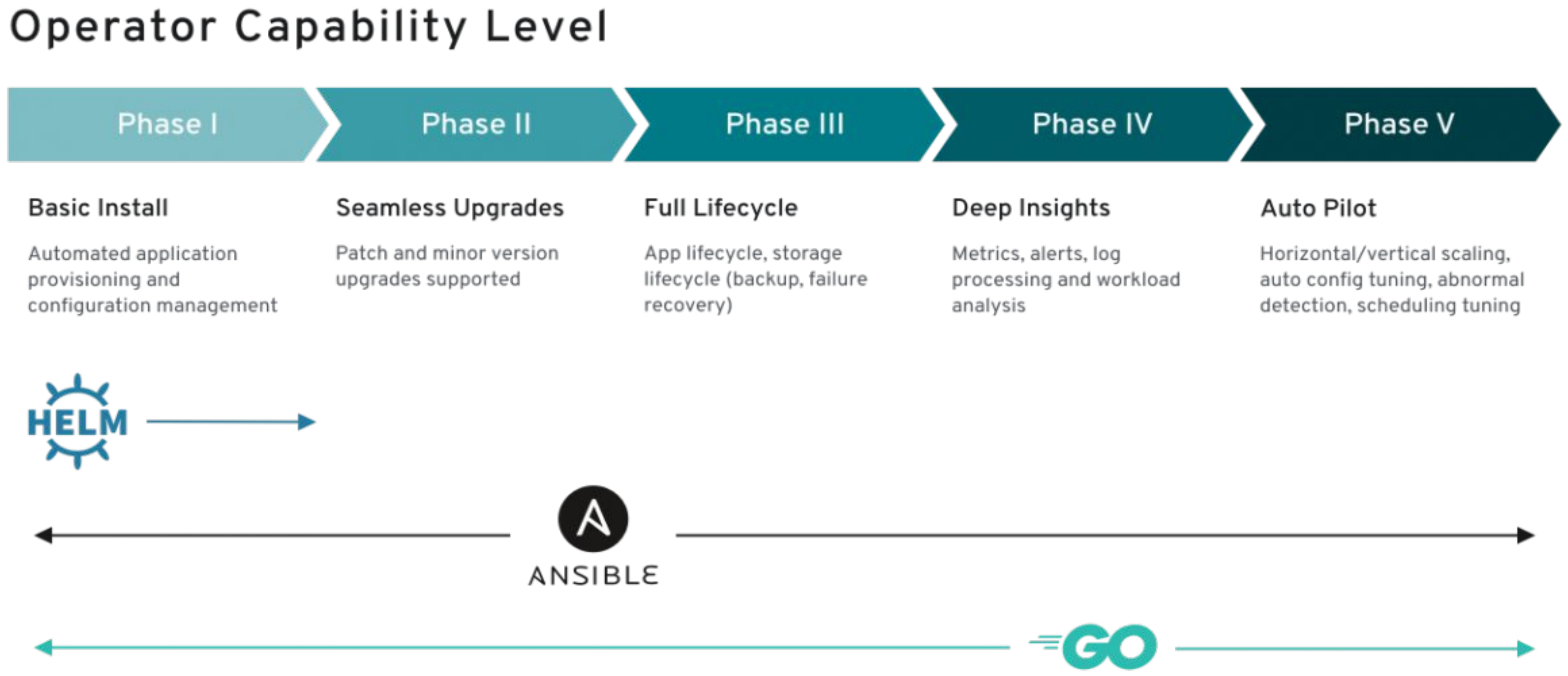
pulp-operator: pulp-operator.git

OPERATOR
FRAMEWORK

Why an operator?

- Kubernetes merely ensures the desired state of the application
- Manifests are static; no variables as input to desired state
  - If you upload a newer version of the same manifest, K8s will adjust the state
- Little flexibility in the desired state at runtime
  - A prominent exception: horizontal scaling
- An operator is a running container that manages variable state for things like upgrades & backups
- User settings in a "custom resource" (cr) yaml (a K8s object)
- A fully-featured operator provides an experience comparable to:
  - An app store app (OperatorHub.io)
  - A managed cloud service

Features of an Operator



Operator Capability Level

Phase I	Phase II	Phase III	Phase IV	Phase V
Basic Install	**Seamless Upgrades**	**Full Lifecycle**	**Deep Insights**	**Auto Pilot**
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Some cool features we have already

- # of pulp-workers & pulp-content instances can be defined ahead of time or manually updated at runtime
- pulp-content instances receive evenly distributed load
- /var/lib/pulp/ can be expanded at runtime (if infra supports expanding it)
- Entire installation (several plugins) happens in only a few minutes

User experience using pulp-operator

- Already have K8s setup
- Clone pulp-operator git repo
- Copy & modify our “custom resource”
  deploy/crds/pulpproject_v1alpha1_pulp_cr.default.yaml ->
  deploy/crds/pulpproject_v1alpha1_pulp_cr.yaml
  - We have a couple of pre-defined cr.yaml files as well
- ./up.sh (uses `kubectl`)
  - pulp-operator image gets downloaded & run
  - pulp-operator pulls in/runs postgres, redis, and pulp images
- State can be viewed graphically using K8s dashboard (WebGUI)

What is defined where?

- pulpcore.git/containers:
  - Static environment variables
  - RPM dependencies
  - Which plugins get installed and from which pip install strings (via variables) or folders
  - Database initialization
  - What scripts/command get run before starting the Pulp services
  - Mapping “pulp-api”, etc. to actual commands
  - Ports used
- Pulp-operator.git
  - /etc/pulp/settings.py
  - postgres
  - redis
  - networking / storage
  - number of containers (instances)
  - Calling “pulp-api”, etc.

Related Tooling

plugin-template (CI)

- Install.sh:
    - Creates pulpcore.git/containers/ vars.yaml
    - Builds “pulp_foo” image
    - Creates operator cr.yaml
    - calls from pulp-operator.git:
        - .travis/k3s-install.sh: Install & configure k3s
        - up.sh: bring up containers
        - .travis/pulp-operator-check-and-wait.sh: Waits till containers come up; checks status page; prints which prior steps failed
- Script.sh:
    - Uses aliases like $CMD_PREFIX to install temporary testing tools into pulp-api container & run unit tests
    - pytest calls pulp-smash; which can now reach into pulp-api container as well

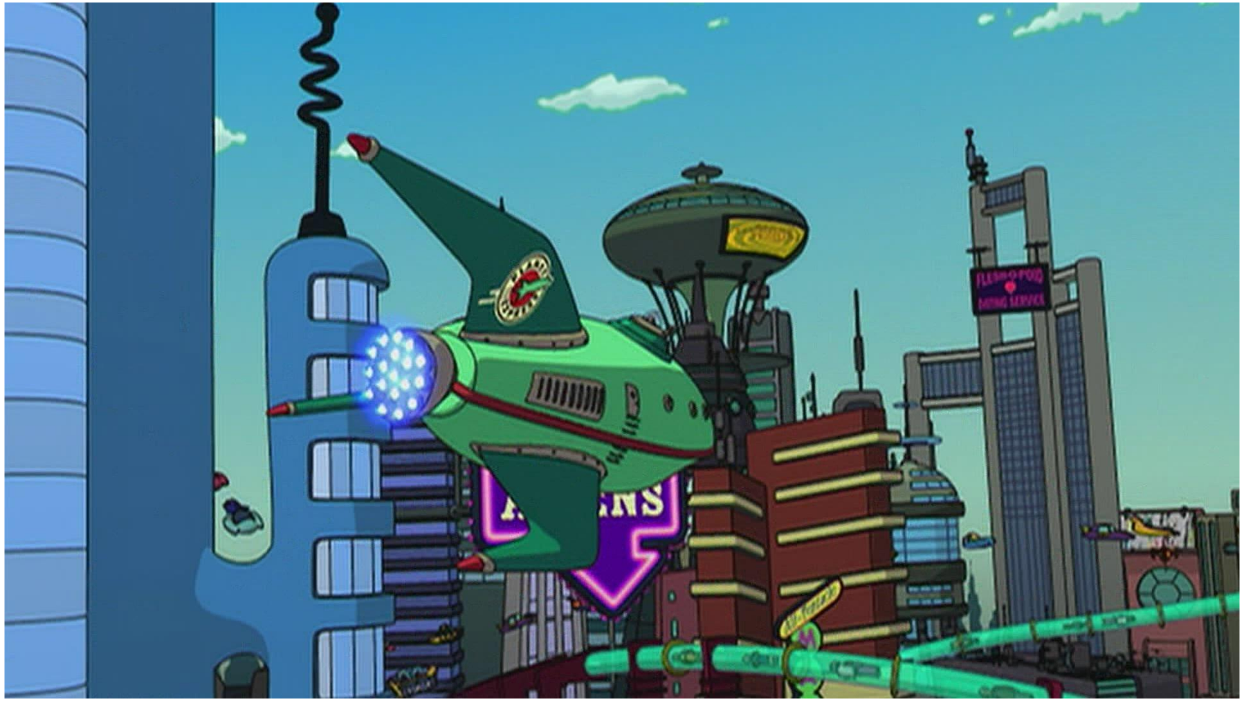
Pulp-operator: pulp-insta-demo.sh

- 103-line wrapper around .travis/k3s-install.sh & up.sh
- Can be downloaded directly; will download pulp-operator git repo
- Configuring system forbidden; very few deps; risk of not working properly
- Travis CI tests on Ubuntu 16.04
- Manually tested via Vagrant on Ubuntu 16.04 / 18.04, CentOS 7 & Fedora
- User experience:
  - Run script & review output
  - k3s uninstall script services as entire uninstaller
- On homepage; blog post to be written

pulp-demo.git

- Specifically meant for demoing pulp at conferences
- After Fedora Workstation is installed on a NUC and accessible via SSH:
    - Installs minikube
    - Installs related tooling like httpie
    - Configures OS; even GNOME shortcut to K8S dashboard
    - You can then run pulp-operator’s ./up.sh

Future Development



Things that should be done the Kubernetes way

- nginx load-balancing
- pulp-settings needs to query the externally accessible hostname
  - `hostname` returns container private network hostname
  - Need new K8s object(s) for externally accessible service

Further CI

- Epic 5393
- pulp_foo image & 7-plugins “pulp” image based on “pulpcore”
- publish images (pulp-operator only one done so far)
- TBD: When to publish 7-plugins “pulp” image?
  - Wait for all 7 plugins to release & succeed?
  - We do not want newer versions of the image name to ever contain fewer content plugins.
- TBD: Versioned releases
  - What if we need to make operator/container changes after code release (like downstream RPMs?)
- TBD: Let plugins provide snippets for a common Dockerfile, beyond just the pip install string
  - Could become unmanageable or incompatible with eachother
  - RPM variable preferred

Highlight of TODO before maturity model Phase 1

- Mostly in Epic 5132 (publish to OperatorHub)
- Need to make it work across a greater % of environments:
  - Mainly: Our K8S-managed storage requirement of “shared filesystem across every node” is incompatible with many K8s clusters’ storage, like Ceph
- pulp-settings needs to query the externally accessible hostname
- Some permissions concerns
- Molecule CI

Getting to phase 2 through 5

- 2 Epics on redmine need to be reworked for these
- The vision: “A kick-ass cluster for pulp”

Special thank you to:

- Eric Helms
  - Starting this entire sub-project
  - Working prototype against Pulp a year ago
- Dennis Kliban
  - Integration into CI over the summer
  - Feature development now
- SysEleven
  - Hosting a large production “metakube” cluster for us

Questions?