inovex

Pulp – Software Repository Management

A brief introduction

Arnold Bechtoldt

Karlsruhe, 12.12.14

About

Arnold Bechtoldt

- IT Systems Engineer at inovex GmbH
- Small to large Open Source Datacenter Management solutions
- High available web-based application services
- Contributing to various Open Source projects

Topics

inovex

Introduction

Preparing Pulp

Using Pulp

Scaling &
High Availability

Introduction

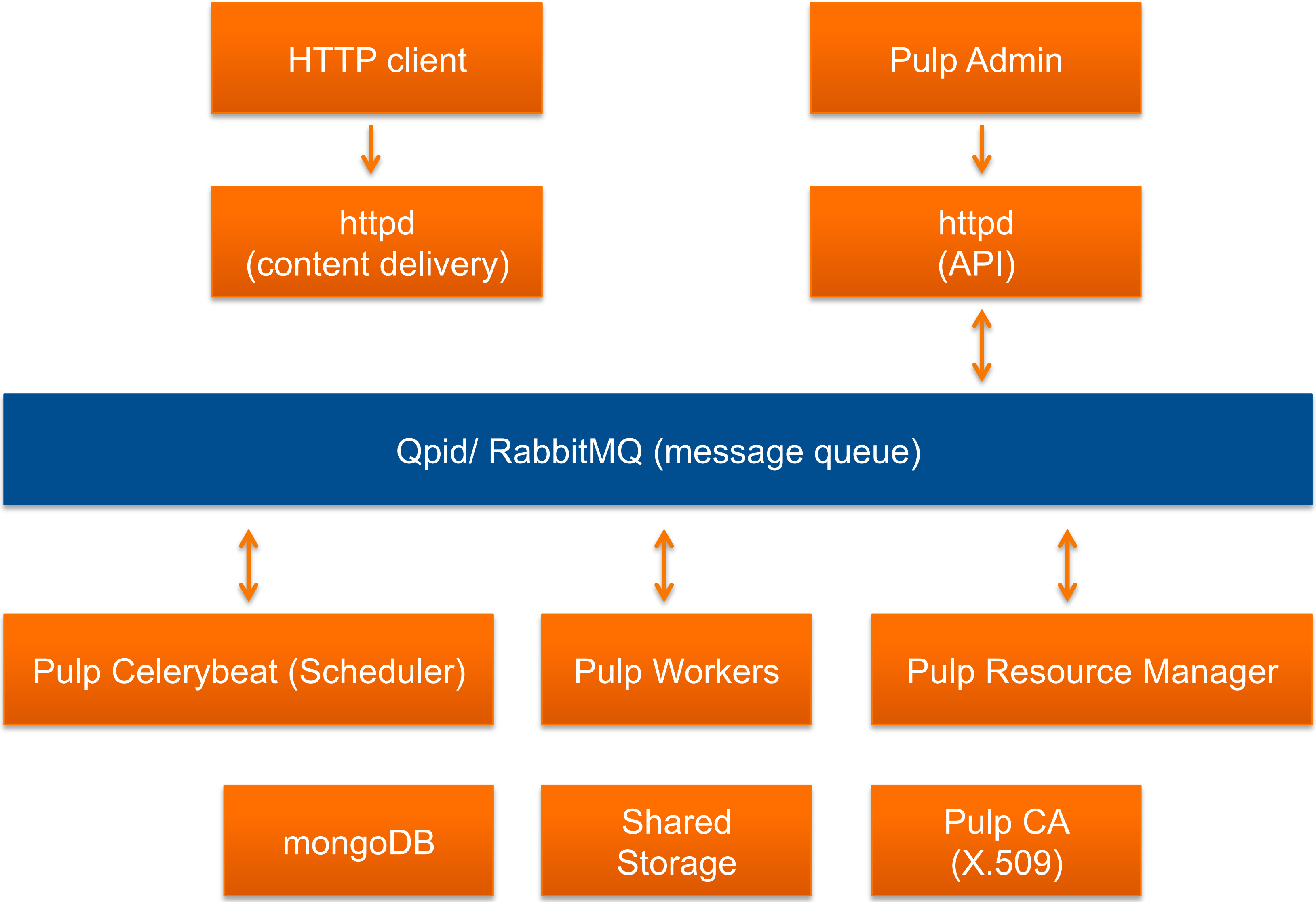
Pulp – Software Repository Management (1/2)

- Provides a platform for RPM repository management
- Built on top of httpd, Celery (task scheduling) and Qpid/ RabbitMQ (message queue)
- Provides a RESTful API and is part of Red Hat Satellite
- Want to setup Pulp with Katello? Check the slides “Katello / Pulp / Candlepin - OSDCM: Repository Management” of Jürgen Brunk at inovex.de

pulpproject.org

Introduction

Pulp – Software Repository Management (2/2)



Preparing Pulp

Preparation

1. Setup a plain CentOS 7 system (e.g. using Vagrant/ VirtualBox)
2. Set correct time settings
3. Disable firewall and SELinux (in your lab only ☺)
4. Install EPEL & Pulp RPM repository sources in /etc/yum.repos.d/
5. Verify repo connectivity with *yum repolist*

See notes/01_preparations.markdown for details

Preparing Pulp

Installation (1/2)

1. Install mongoDB and Qpid message broker packages

2. Configure and setup mongoDB and Qpid

3. Install Pulp server and Pulp admin packages

4. Verify mongoDB and Qpid are running

5. Stop httpd and migrate Pulp mongodDB with *pulp-manage-db*

6. Start pulp_workers, httpd, pulp_celerybeat and pulp_resource_manager

See notes/02_install.markdown for details

Preparing Pulp

Installation (2/2)

7. Verify all services are running

8. Disable SSL verification in Pulp admin configuration (in your lab only ☺)

9. Verify login works with admin:admin using *pulp-admin*

See notes/02_install.markdown for details

Using Pulp

Mirror an existing repository

1. Create a repository:

   ```
   pulp-admin rpm repo create \
                  --repo-id=foreman \
                  --feed=http://yum.theforeman.org/releases/1.1/el6/x86_64/ \
                  --relative-url=foreman/ \
                  --serve-http=true \
                  --serve-https=true
   ```

2. Sync from upstream:

   ```
   pulp-admin rpm repo sync run \
                  --repo-id=foreman
   ```

3. Verify existence: *http://127.0.0.1/pulp/repos/foreman/*

See notes/03_managing_repos.markdown for details

Using Pulp

Uploading packages (1/2)

1. Create a new repository:

```
  pulp-admin rpm repo create \
                  --repo-id=mycustomrepo \
                  --relative-url=custom/repo/ \
                  --serve-http=true \
                  --serve-https=true
```

2. Start initial publish:

```
  pulp-admin rpm repo publish run \
                  --repo-id=mycustomrepo
```

3. Verify existence: *http://127.0.0.1/pulp/repos/custom/repo/*

See notes/03_managing_repos.markdown for details

Using Pulp

Uploading packages (2/2)

4. Prepare packages to upload:

```
yum install --downloadonly --downloaddir=/var/tmp/saltpkgs/ salt-minion
```

5. Upload packages:

```
pulp-admin rpm repo uploads rpm \
          --repo-id=mycustomrepo \
          --dir=/var/tmp/saltpkgs/
```

6. Publish uploaded packages:

```
pulp-admin rpm repo publish run \
          --repo-id=mycustomrepo
```

7. Verify existence of new packages: *http://127.0.0.1/pulp/repos/custom/repo/*

See notes/03_managing_repos.markdown for details

Using Pulp

Retreiving information about repositories

1. List repositories with their names only:

   pulp-admin repo list -s

2. List repositories with details:

   pulp-admin repo list -s

3. List repositories with even more details:

   pulp-admin repo list --details

See notes/03_managing_repos.markdown for details

Using Pulp

Working with content units (1/2)

1. Search for RPM packages:

```
  pulp-admin rpm repo content rpm \
                 --repo-id=mycustomrepo \
                 --match='name=salt.*’
```

2. Copy RPM packages to another repo:

```
  pulp-admin rpm repo copy rpm \
                 --from-repo-id=foreman \
                 --to-repo-id=mycustomrepo \
                 --match='name=foreman-libvirt.*’
```

3. Publish new packages:

```
  pulp-admin rpm repo publish run \
                 --repo-id=mycustomrepo
```

See notes/03_managing_repos.markdown for details

Using Pulp

Working with content units (2/2)

4. Remove RPM packages:

```
pulp-admin rpm repo remove rpm \
            --repo-id=mycustomrepo \
            --match='name=foreman-libvirt.*’
```

5. Publish changes:

```
pulp-admin rpm repo publish run \
            --repo-id=mycustomrepo
```

6. Verify absence of removed packages: *http://127.0.0.1/pulp/repos/custom/repo/*

See notes/03_managing_repos.markdown for details

Scaling & HA

Pulp in Production

- Pulp workers need several Gigabytes of memory to collect & work with metadata
- Pulp API, resource manager, task scheduler and message queue are lightweight
- Setting up a redundant mongoDB replica set is a good idea
- Deploy additional Pulp worker systems to distribute work load of long running tasks (Sync/ Publish)
- HTTP(S) loadbalancing in front of Apache httpd (delivery/ API)
- Caching of static content units (RPM packages) *could* be useful

We are hiring!

inovex.de

inovex

exciting projects • great technologies • nice colleagues • cool offices

We have excellent job offers in Karlsruhe, Cologne, Munich and Pforzheim!

Thank You!

Questions?

Contact

Arnold Bechtoldt
IT Engineering & Operations

inovex GmbH
Office Karlsruhe
Ludwig-Erhard-Allee 6
D-76139 Karlsruhe

arnold.bechtoldt@inovex.de