



Lecture 3:

BOSH release in-depth



What is a package?

- A Package provides executables, source code and dependencies to jobs
- It should not be tied to a particular stemcell and should not depend on internet access
- Never depend on the presence of libraries or other software on stemcells
- It is composed of source code and/or a blob and instructions on how to build it

```
→ packages git:(master) x tree redis-server
redis-server
├─ packaging
└─ spec
```

Package's spec file

```
name: redis-server
```

← Package name

```
dependencies: []
```

← Package's dependencies (at compile time)

```
files:
```

```
- redis/redis-2.8.3.tar.gz
```

← The location where BOSH can find the binaries and other files that the package needs at compile time.

BOSH interprets the locations you record in the files section as being relative either in the “src” directory or in the “blobs” directory

Package's packaging script

```
set -e # exit immediately if a simple command exits with a non-zero status
set -u # report the usage of uninitialized variables

# Available variables
# $BOSH_COMPILE_TARGET - where this package & spec'd source files are available
# $BOSH_INSTALL_TARGET - where you copy/install files to be included in package
export HOME=/var/vcap

tar xfv $BOSH_COMPILE_TARGET/redis/redis-2.8.3.tar.gz
cd redis-2.8.3
make PREFIX=${BOSH_INSTALL_TARGET} install
```

Ensure that any copying, installing or compiling delivers resulting code to the install target directory

Package's dependency graph

Be aware that BOSH ensures that dependencies cited in the dependencies block of package spec files are available to the deployed binary, but only at compile time

```
---  
name: ruby_1.9.3  
  
dependencies:  
- libyaml_0.1.4
```

```
set -e -x  
  
tar xzf ruby_1.9.3/ruby-1.9.3-p484.tar.gz  
pushd ruby-1.9.3-p484  
./configure \  
  --prefix=${BOSH_INSTALL_TARGET} \  
  --disable-install-doc \  
  --with-opt-dir=/var/vcap/packages/libyaml_0.1.4  
  
make  
make install  
popd
```

Sources

- Provides packages with the non-binary files they need
- We store them at the “src” directory
- Use the globbing pattern `<package_name>/**/*` to deep-traverse the directory in src where the source code should reside

```
files:  
- myappsrc/**/*
```

Blobs

- Packages often use tar files or other binaries, also known as blobs
- Checking blobs into a repository is problematic if your repository is unsuited to dealing with large binaries (as is true of Git, for example)
- We store them in the “blobs” directory

Configuring a blobstore

In the `config` directory, you record the information BOSH needs about the blobstore:

- The `final.yml` file names the blobstore and declares its type, which is either local or one of several other types that specify blobstore providers.
- The `private.yml` file specifies the blobstore path, along with a secret. It contains keys for accessing the blobstore and should not be checked into a repository.

final.yml example for AWS S3 blobstore

```
---  
final_name: redis  
blobstore:  
  provider: s3  
  options:  
    bucket_name: redis-boshrelease
```

← Release name

← Blobstore type

← Specific blobstore configuration

private.yml example for AWS S3 blobstore

```
---
blobstore:
  s3:
    access_key_id: <AWS ACCESS KEY ID>
    secret_access_key: <AWS SECRET ACCESS KEY>
```

← Specific blobstore configuration

NEVER check this file into a repository.

What is a job?

A job describe pieces of the service or application you are releasing:

- What packages depends on
- How to start/stop it
- How to monitor it
- What properties can be configured

Elements of a Job

- Spec: describes the job specification (package's dependencies, templates and properties)
- Monit: describes how to run the job's processes
- Templates: control scripts, configuration files, ...

Spec file

```
---
name: redis
packages:
- redis-server
templates:
  bin/redis_ctl: bin/redis_ctl
  bin/health_check: bin/health_check
  bin/monit_debugger: bin/monit_debugger
  bin/initialize_kv: bin/initialize_kv
  data/properties.sh.erb: data/properties.sh
  helpers/ctl_setup.sh: helpers/ctl_setup.sh
  helpers/ctl_utils.sh: helpers/ctl_utils.sh
  config/redis.conf.erb: config/redis.conf
```

← Job name

← Job's package dependencies

← Job's templates

Spec file

```
properties:
  redis.port:
    description: Port to listen for requests to redis server
    default: 6379
  redis.password:
    description: Password to access redis server
    default: "r3d!s"
  redis.master:
    description: IP address or hostname of the Redis master node
```

Property name

Property description

Property default value

Monit file

```
check process redis
  with pidfile /var/vcap/sys/run/redis/redis.pid
  start program /var/vcap/jobs/redis/bin/redis_ctl start
  stop program /var/vcap/jobs/redis/bin/redis_ctl stop
  group vcap
```

Process name

How to monitor the process

Ownership of the process

How to start/stop the process

Templates

```
#!/bin/bash

set -e # exit immediately if a simple command exits with a non-zero status
set -u # report the usage of uninitialized variables

# Setup env vars and folders for the webapp_ctl script
source /var/vcap/jobs/redis/helpers/ctl_setup.sh 'redis'

export LANG=en_US.UTF-8

echo redis redis $PIDFILE $(cat $PIDFILE)

case $1 in
  start)
    pid_guard $PIDFILE $JOB_NAME

    exec redis-server /var/vcap/jobs/redis/config/redis.conf
    ;;

  stop)
    redis-cli shutdown
    kill_and_wait $PIDFILE

    ;;

  reload)
    echo redis reloading...
    $JOB_DIR/bin/redis_ctl stop
    $JOB_DIR/bin/redis_ctl start

    ;;

  *)
    echo "Usage: redis_ctl {start|stop}"

    ;;
esac
exit 0
```


Special scripts

- **bin/pre-start**: will run before the job is started. This script allows the job to prepare machine and/or persistent data before starting its operation.
- **bin/drain**: will run when the job is restarted or stopped. This script allows the job to clean up and get into a state where it can be safely stopped
- **bin/run**: will run a job's errand