vmware®

Concourse

CI that scales with your project

Continuous Integration & Delivery

Benefits

AUTOMATION.
Integrate tools and automate processes from testing to builds and deployment

SPEED.
Release more frequently with smaller bits will reduce complexity and improve time-to-market

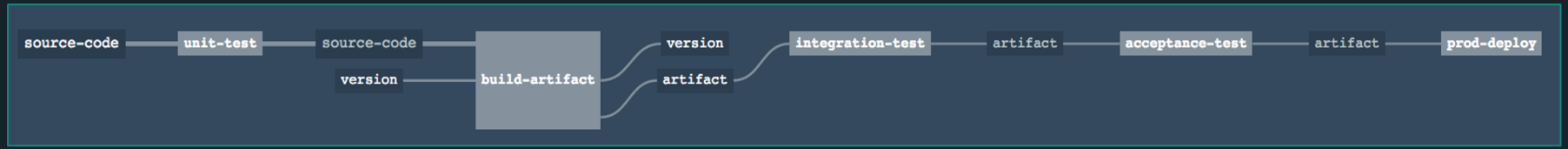
QUALITY.
Reduce feedback loop using test-driven development to surface problems sooner and be responsive

AGILITY.
Push updates on regular basis with no downtime to improve customer experience and time to market

Concepts

Commit Code Change → **Automate Build & Test** → **Store Binaries & Build Artifacts** → **Automated Integration Testing** → **Acceptance, Performance & Load** → **Zero Downtime Upgrade to Production**

Pipeline



vmware

What we found in other CI systems

Snowflakes
- lots of plugins
- system dependencies
- textbox scripting

Pipelines
- no first-class support
- complex job sequencing

Environment Parity
- works locally, breaks on server
- lots of debugging commits

Usability
- complicated UIs
- endless menus
- too many clicks to get logs

Execution Hierarchy
- deep and complex

Scalability
- hard to scale vertically or horizontally

Concourse Principles

Simple

Concourse is a response to the complexity introduced by other systems. It is built on the idea that the best tools can be learned in one sitting.

Usable

Concourse is optimized for quickly navigating to the pages you most care about. From the main page, a single click takes you from a pipeline view to the log of a job's latest failing build.

Isolated Builds

Every build task is executed in a container defined by its own configuration, by stateless workers. This eliminates build pollution and ensures multiple teams can use the same Concourse deployment without worrying about the state of the worker VMs.

vmware

Concourse Principles

Scalable, reproducible deployment

No Concourse deployment is a snowflake. There are no boxes to check; no configuration happens at runtime.

Flexible

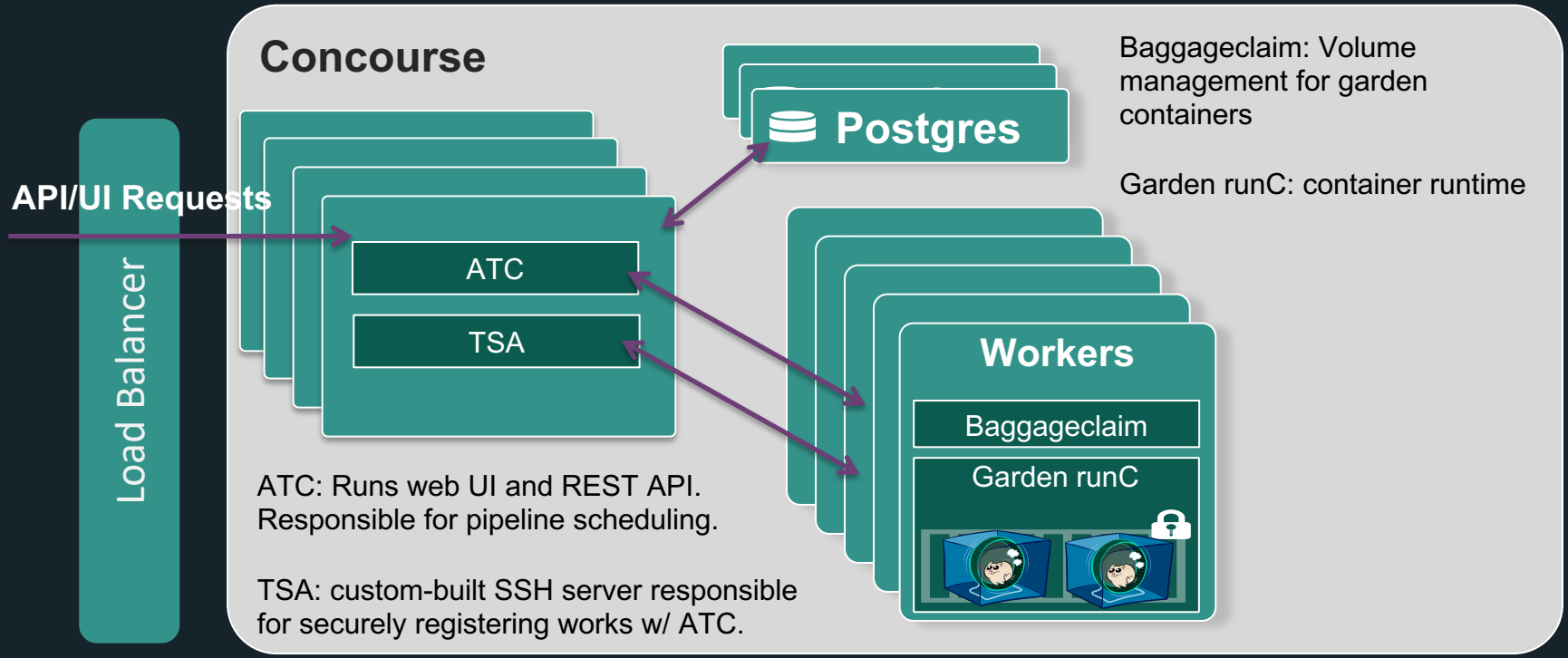
Features that other systems implement in the core of the product, Concourse implements in "userland", as resources. This keeps the core of Concourse small, simple, and proves out the extensibility introduced by this simple interface.

Local iteration

Concourse supports running one-off builds from local task configuration that allows you to trust that your build running locally runs exactly the same way that it runs in your pipeline.

vmware

Concourse Architecture



Concourse Concepts: simple primitives

Resources

detecting, fetching, creating of external versioned “things”

```
# pipeline.yml
resources:
- name: pcfdemo
  type: git
  source:
    uri: https://github.com/.../PCF-demo.git
    branch: master
```

- encapsulation of some external resource
- replaces plumbing scripts
- results in intuitive pipeline semantics
- many first-class concepts from other systems are implemented in terms of resources (ex: timed triggers)
- only pluggable interface

git repo, s3 bucket, docker image, bosh deployment, bosh.io release, bosh.io stemcell, pivnet

Pivotal Tracker, github release, cf, vagrant cloud/atlas, time, semver, http

Concourse Concepts: simple primitives

Tasks

run a script in a container with its dependent inputs

```
# unit.yml
platform: linux
image: docker:///java#8
inputs:
  - name: pcfdemo
run:
  path: mvn
  args: [ clean, test ]

OR

run:
  path: program.sh

# program.sh
#!/bin/bash
echo "do something…"
```

- Execution of unit of work in isolated environment i.e. in a container
- All tasks executed separately from each other
- Typically contains definition of:
  - platform
  - base container image
  - inputs and outputs
  - command or script to execute

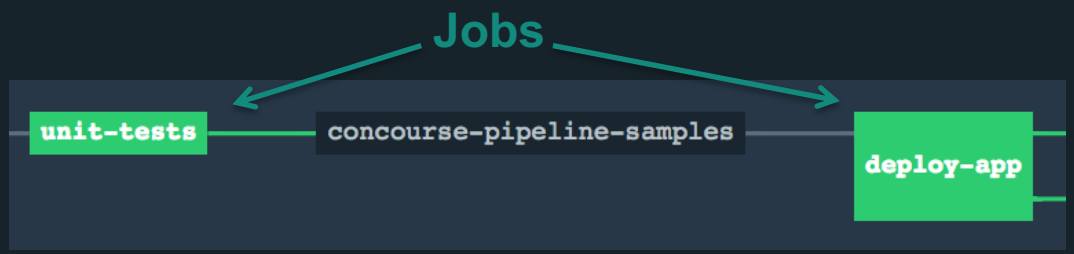
Concourse Concepts: simple primitives

Jobs

compose resources and tasks together to do something (run tests, ship, etc.)

```
# pipeline.yml
jobs:
- name: QA
  plan:
  - get: pcfdemo
    trigger: true
  - file: pcfdemo/unit.yml
```

- Describes a set of actions to perform in a Build Plan
- Build Plan defines tasks, sequencing, success/failure triggers, upstream triggers, timeouts, retries, etc
- Individual execution of Job is a Build



Concourse Concepts: pipelines

```yaml
# pipeline.yml
resources:
- name: pcfdemo
  type: git
  source:
    uri: https://github.com/.../PCF-demo.git
    branch: master

jobs:
- name: QA
  plan:
  - get: pcfdemo
    trigger: true
  - task: do-something
    file: pcfdemo/unit.yml
```

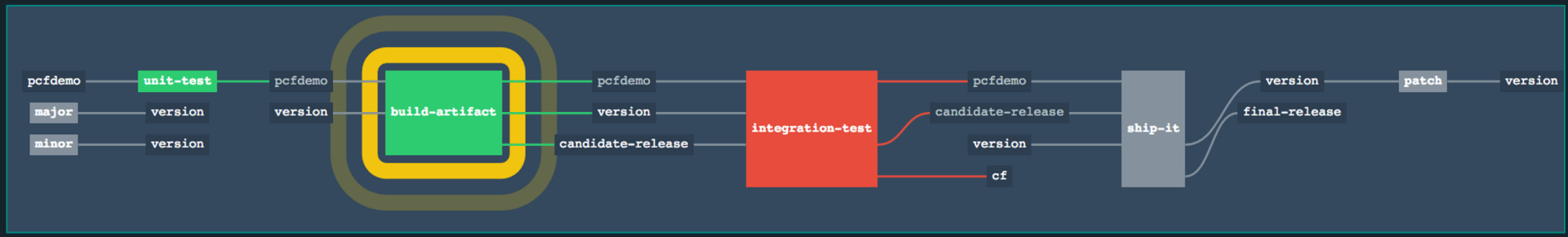
```yaml
# unit.yml
platform: linux
image: docker:///java#8
inputs:
  - name: pcfdemo
run:
  path: mvn
  args: [ clean, test ]

OR

run:
  path: program.sh

# program.sh
#!/bin/bash
echo "do something…"
```

Concourse Concepts: pipelines

the resulting flow of resources through jobs

visualization UI for build monitor

many isolated pipelines per deployment



fly execute: run task with local bits

```
~/git/PCF-demo » fly execute -c ci/tasks/build.yml -i pcfdemo=.
executing build 92
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 58.8M    0 58.8M    0     0  28.1M      0 --:--:--  0:00:02 --:--:-- 28.1M
initializing with docker:///java#8
running pcfdemo/ci/tasks/build.sh
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building pcf-demo 1.0.0-BUILD-SNAPSHOT
[INFO] ------------------------------------------------------------------------
...
[INFO] Packaging webapp
[INFO] Assembling webapp [pcf-demo] in [/tmp/build/e55deab7/pcfdemo/target/pcfdemo]
[INFO] Processing war project
[INFO] Copying webapp resources [/tmp/build/e55deab7/pcfdemo/src/main/webapp]
[INFO] Webapp assembled in [61 msecs]
[INFO] Building war: /tmp/build/e55deab7/pcfdemo/target/pcfdemo.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 33.423 s
[INFO] Finished at: 2016-02-03T12:56:54+00:00
[INFO] Final Memory: 20M/217M
[INFO] ------------------------------------------------------------------------
succeeded
```

fly hijack: hop into build's container

```
~/git/PCF-demo » fly hijack -j pcfdemo/build-artifact
1: build #10, step: version, type: get
2: build #10, step: prepare-build, type: task
3: build #10, step: candidate-release, type: put
4: build #10, step: version, type: put
5: build #10, step: pcfdemo, type: get
6: build #10, step: version, type: get
7: build #10, step: candidate-release, type: get
8: build #10, step: build, type: task
choose a container: 2
root@bqvgog0t9s0:/tmp/build/5020c204# ls -al
total 8644
drwxr-xr-x 1 root root      84 Feb  3 13:14 .
drwxr-xr-x 1 root root      16 Feb  3 13:14 ..
drwxr-xr-x 1 root root      14 Feb  3 13:14 build
-rw-r--r-- 1 root root 8849783 Feb  3 13:14 pcf-demo-1.1.0-rc.4.war
drwxr-xr-x 1 root root     242 Feb  3 11:24 pcfdemo
drwxr-xr-x 1 root root      12 Feb  3 13:14 version
root@bqvgog0t9s0:/tmp/build/5020c204# echo `cat version/number`
1.1.0-rc.4
root@bqvgog0t9s0:/tmp/build/5020c204#
```

fly set-pipeline: iterate on pipeline

```
~/git/PCF-demo » fly set-pipeline -p pcfdemo -c ci/pipeline.yml -l ~/.concourse/pcfdemo-properties.yml
resources:
  resource cf has changed:
    name: cf
    type: cf
    source:
      api: https://api.local.micropcf.io
      organization: micropcf-org
      password: admin
      skip_cert_check: true
      skip_cert_check: false
      space: micropcf-space
      username: admin

apply configuration? [yN]:
```

vmware®

Open.
Agile.
Cloud-Ready.