Deployment

G. Molines

2017-2018

TEST SET-UP

How do you launch a test?

What happens when you type mvn test?

How do you launch a test?

What happens when you type mvn test?

→Maven forks a jvm to launch tests

→Everything is run in a single command line

→Your tests need be self contained

→Can’t really launch a distributed test

How do you test client-server?

How do you test client-server?

- Launch server
- **Deploy** target app
  - Let’s call it “System Under Test”, aka SUT
- Launch client test:
  - mvn test client side

App changed

- → need to **redeploy** app
  - `cp myapp.ear $TOMCAT_HOME/webapps`

- Launch client test:
  - mvn test client side

Deployment

- = make software available for use
  - Copy packages
    - Install, binary copy
  - Bind resources
    - to OS
    - to container, Eg: app server

Cold deploy

- How do you test an installer?
- OS adherence
- Uninstall: does it leave the system in same state?
- OS or server restart
- Long process

Hot deploy

- Without OS / server restart
  - Continuity of service
- Useful during development phase
- Never guaranteed to work fine
  - Left-over
  - Memory usage
  - Would it work the same with cold deploy?

Link with CI

Handling distributed tests in CI

- Deployment needs be automated
  - Hot
  - scripted
  - remoteable
    - Deal with network, not files
- How to run in the server?
- How to get the results?

Data

- Tests need be repeatable
  - Same data start state
  - Need to setup the start state
- Setting up distributed env is expensive / long

Data

- Data in dedicated DB?
  - Need be loaded upfront
  - Test interferences
- → Loading data **is** part of automation

- Data cleaning also

Data

- Customer data
  - Manage, gather, keep current, tie to versions
  - Anonymise
  - Volume, how to split?

Configurations

Testing env. variation

- Need to execute the **same** test, but vary
  - The OS
  - OR, the DB
  - OR, the browser
  - Etc.
- → your tests should be independent from env. as much as possible

Environmental changes

- OS, DB, browser, JDK version, JDBC driver type, screen size, …
  - Combinatorial explosion
- Platform

coverage

How to deploy

Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

Approach #1

- Setup environment
  - DB
    - Pre-loaded with
  - App server
    - Inc. resources
  - Dependents
- Copy app ear file in app server
- Launch client test

Approach #2

- Setup environment
  - DB
    - Pre-loaded with
  - App server
    - Inc. resources
  - Dependents
- Copy app ear file in app server SCRIPT
- Launch client test SCRIPT

Approach #3

- Setup environment
  - DB
    - Pre-loaded with

SCRIPT

  - App server
    - Inc. resources
  - Dependents
- Copy app ear file in app server

SCRIPT

- Launch client test

SCRIPT

Approach #4

- Setup environment
  - DB
    - Pre-loaded with SCRIPT
  - App server
    - Inc. resources SCRIPT
  - Dependents
- Copy app ear file in app server SCRIPT
- Launch client test SCRIPT

Approach #5

- ~~Setup~~ Build environment
  - DB
    - Pre-loaded with
  - App server
    - Inc. resources
  - Dependents
- Copy app ear file in app server
- Launch client test

Image:
- **Zip file**
- **VM**
- **docker**

Approach #6

- ~~Setup~~ Build environments
  - DB
    - Pre-loaded with
  - App server
    - Inc. resources
  - Dependents
- Copy app ear file in app server
- Launch client test

Approach #7

- Need test execution orchestration to:
  - Allocate env. Dynamically
  - Can scale
  - Collect test results
  - Can preserve machine state

Test orchestration

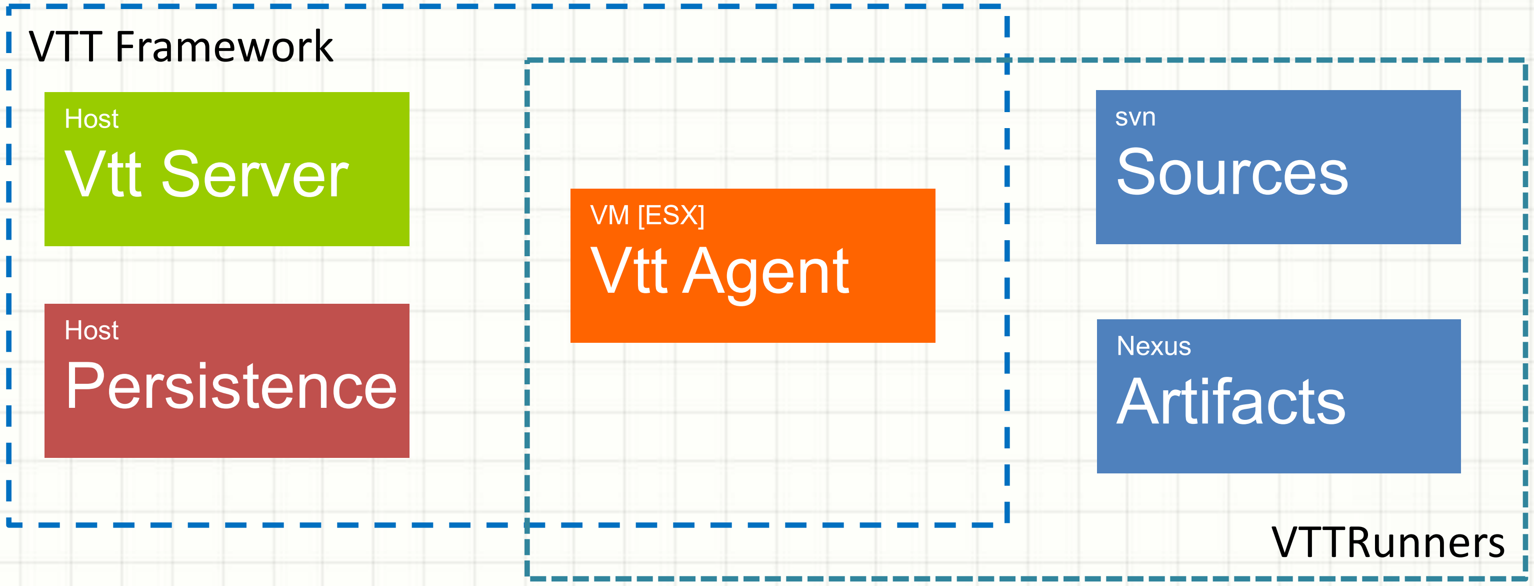
- One test env. can be several machines
  - Sync them on startup / readyness state
  - Each with a role
    - Eg: DB, server, test client
  - Need image to build each machine

- Also: SUT can be several nodes
  - DB, node1, node2, … node n, test client

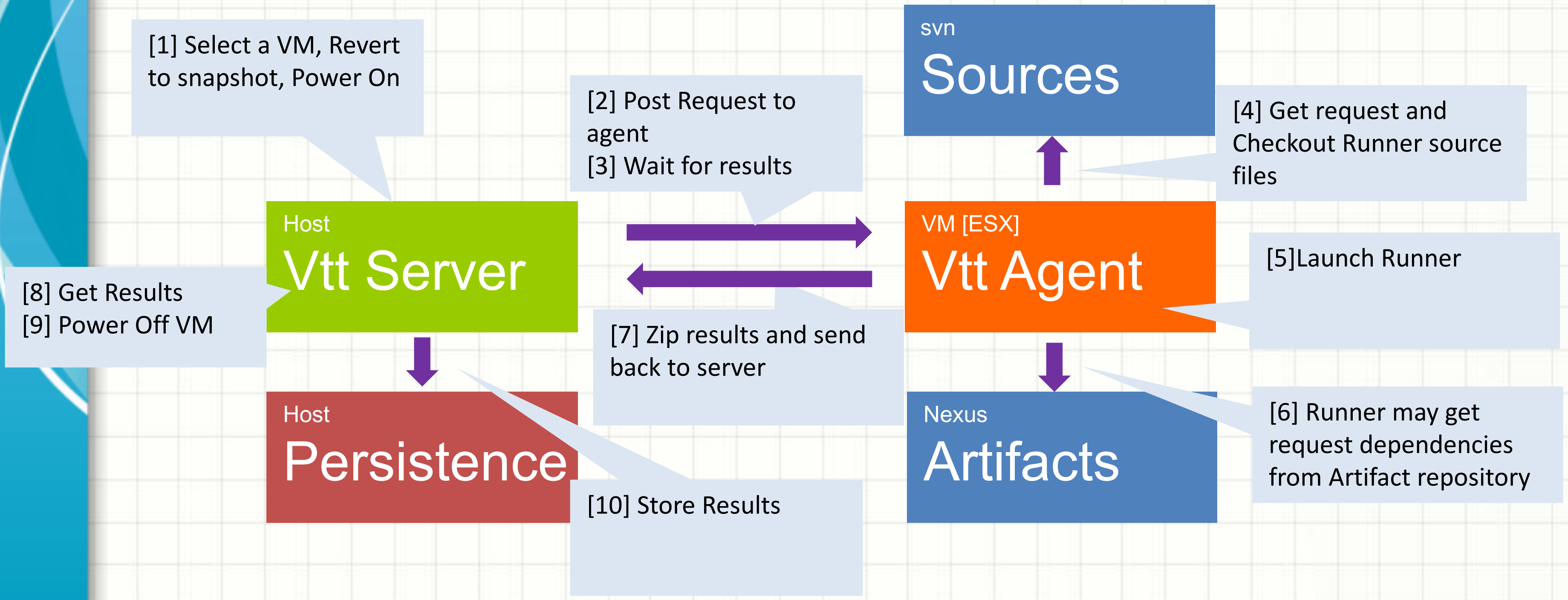
Demo

High Level Architecture

- VTT Framework ➔ Schedule campaign, manage machines, Control agent, Store and report test results
- VTT Runners ➔ Executed by Vtt Agent, setup from sources and artifact dependencies, provide test results



Concept of operation


[1] Select a VM, Revert to snapshot, Power On
[2] Post Request to agent
[3] Wait for results
svn
Sources
[4] Get request and Checkout Runner source files
Host
Vtt Server
VM [ESX]
Vtt Agent
[5]Launch Runner
[8] Get Results
[9] Power Off VM
[7] Zip results and send back to server
Host
Persistence
Nexus
Artifacts
[6] Runner may get request dependencies from Artifact repository
[10] Store Results

QUESTIONS?

APPENDIX