

OSGi WITH DOCKER

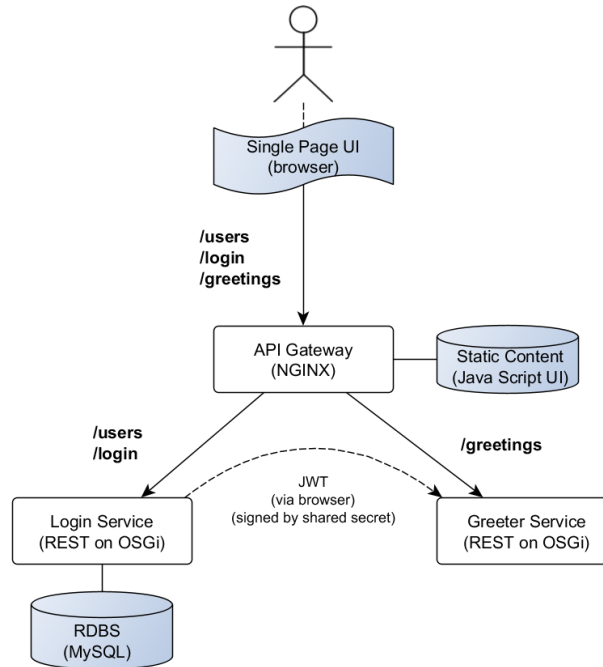
A POWERFUL WAY TO DEVELOP JAVA SYSTEMS

Udo Hafermann
Todor Boev

OVERVIEW

- In this demo we present a tool chain from classes, to bundles, to containers, to systems
- OSGi and Docker come together providing “modularity in the small” and “modularity in the large”
- Grow distributed systems on your local machine and test them with plain JUnit at all levels of granularity - classes to systems
- Update the system without container rebuilds
- Increase your productivity but also gain new insights

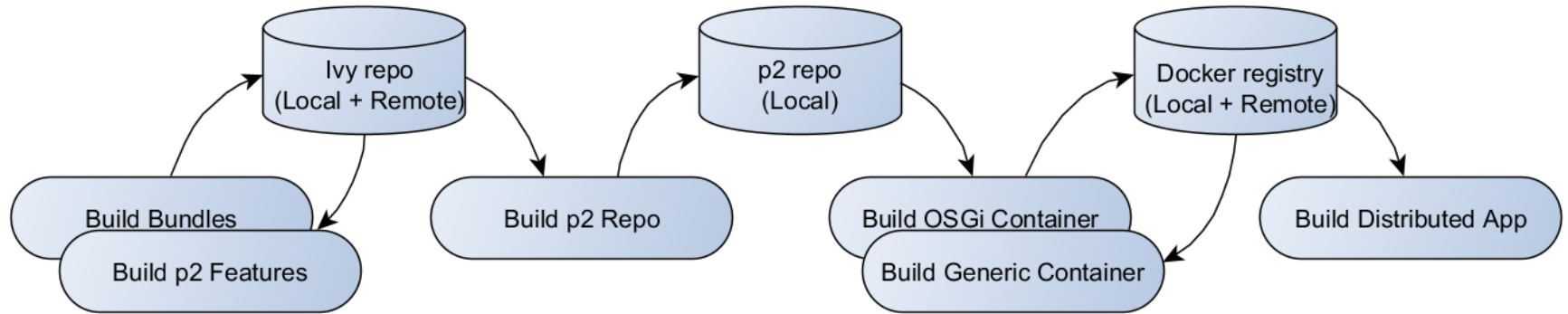
SAMPLE APPLICATION



COMPOSITION LAYERS

- Bundles implementing the functionality (services etc.)
 - Collected in a repo
 - Compile or provision
- Features defining higher-level components
 - Collected in a repo
- Runtimes (OSGi and non-OSGi)
 - Provisioned from higher-level components
 - Collected as container images ready for standing up systems
- Systems
 - Configured and instantiated from images

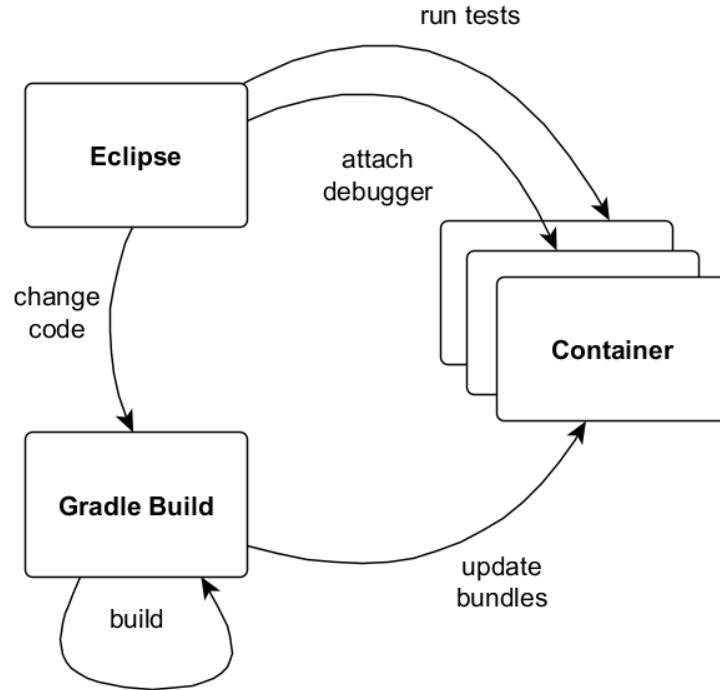
BUILD OVERVIEW



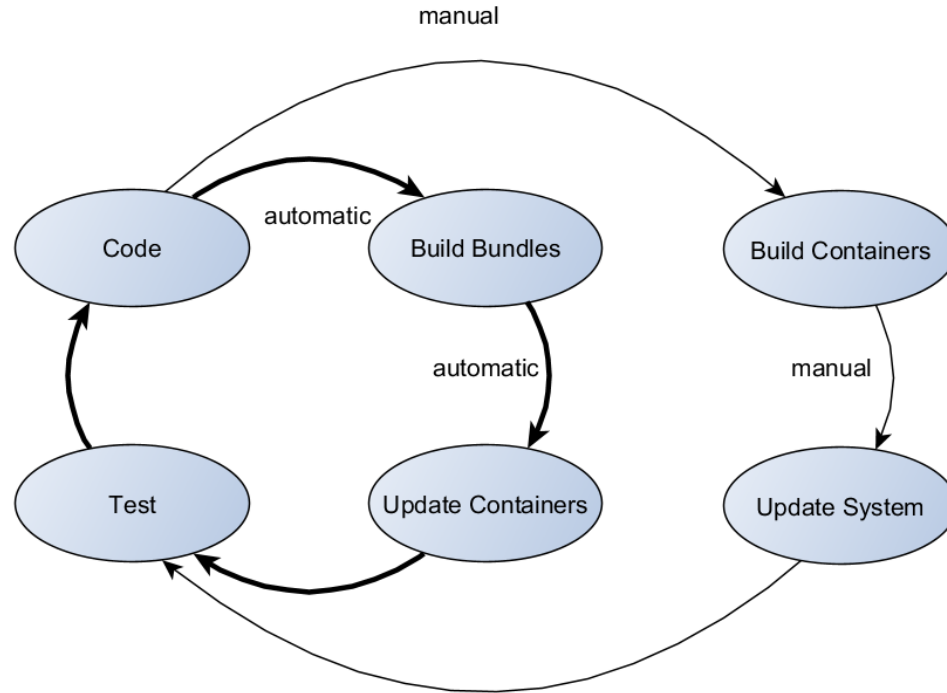
DEMO: BUILD THE SYSTEM

- Run (local) Gradle build to create bundles, features, containers
- Use docker-compose to stand up the system

EDIT-COMPILE-RUN



DEVELOPMENT CYCLES



DEMO: DEV CYCLES

- Running “external” system test
- Attach to a system’s (OSGi) console
- Live update of a bundle
- Attach a debugger
- Deploy and run a test from inside a container

FINDINGS

- Suitable modularity at all levels
 - OSGi gives us flexibility/freedom to compose logic rather than containers
 - Component model independent of container model
- Flexibility in the “middle layer”
- Containers can be used for dev and production
- Developers can work with the OSGi runtime representation in the running system
 - Dynamic updates, tests
- Efficiency
 - Minimal container footprint
 - Quick turnaround
 - End-to-end (JUnit) testing

OPEN ENDS

- Limitations of the example
 - Persistence
 - Production deployment, orchestration (Swarm, Kubernetes)
- Too much configuration
 - E.g. ports for testing
 - Tooling gateway
 - Discovery

 **software** ^{AG}

