
WildFly Glow
An evolution of WildFly provisioning
jdenise@redhat.com
WildFly

Agenda

- What is WildFly provisioning
- How WildFly Glow boosts the user experience
- WildFly Glow in action, demos!
- Current status

You are going to learn how to efficiently produce trimmed WildFly server to run your applications

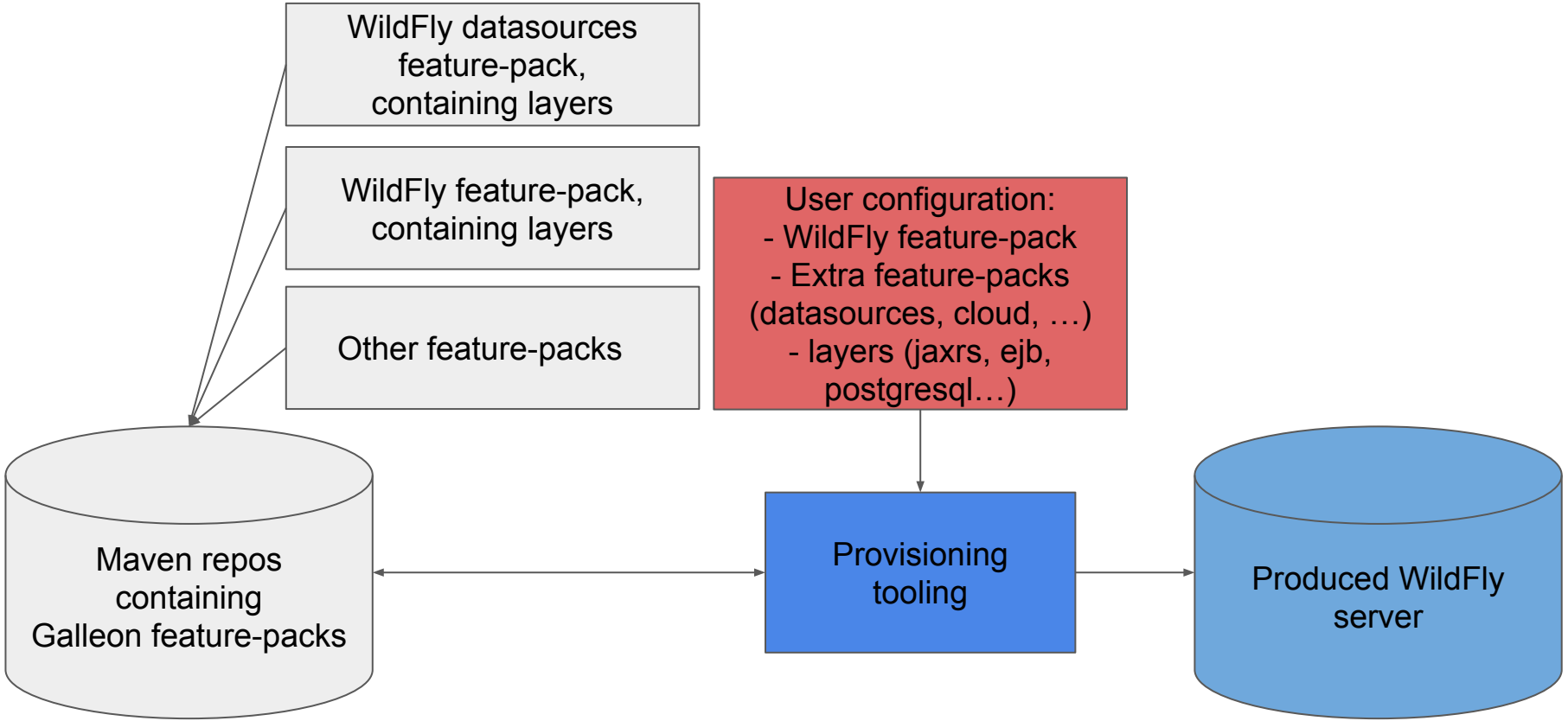
What is WildFly provisioning ?

- Ability to create a WildFly server installation on the fly
- Ability to choose the set of features you want to see present in the created server
- Ability to extend the capabilities of WildFly server (e.g.: Datasources, keycloak SAML)
- The produced server has a smaller size than a full installation, smaller memory footprint
- Benefits: resource consumption, smaller attack surface, simpler server configuration

How WildFly Provisioning is operated ?

- Galleon is the technology on which WildFly provisioning is based
  - Feature-packs: Server metadata container
  - Layers: A server feature/API (e.g.: jaxrs, jsf, ejb, ...)
- Provisioning comes with some tools
  - Command Lines: Galleon CLI
  - Maven Plugins: Galleon Maven Plugin, WildFly Maven Plugin, WildFly Bootable JAR Maven Plugin

Provisioning workflow



Issues with current WildFly provisioning

- Mainly at the user provisioning configuration level
- How to discover WildFly compatible Galleon feature-packs?
- How to discover the Galleon layers that my application requires to properly work?
- Today use documentation, search for blogs and/or github projects to discover extra feature-packs and combination of layers

How to fix them ?

- We need a bridge between the deployed application(s) and the provisioned server
- This is what WildFly Glow is offering, a bridge between the deployment and the server to provision

WildFly Glow

- Glow stands for “Galleon Layers Output from War”
  - Just a name, it also supports jar and ear.
- By scanning the deployment, it can produce the set of Galleon feature-packs and Layers that your application requires
- Documentation reachable from WildFly Documentation

WildFly Glow Features (1/2)

- Understands the connection that exists between Galleon layers and your application
    - Java types and annotations in use
    - XML descriptors,
    - Properties files, ...
- Can suggest interesting features not directly required by your application but meaningful: SSL, Microprofile OpenAPI, WildFly CLI
- Can identify errors and suggest you ways to fix them (eg: missing datasource)

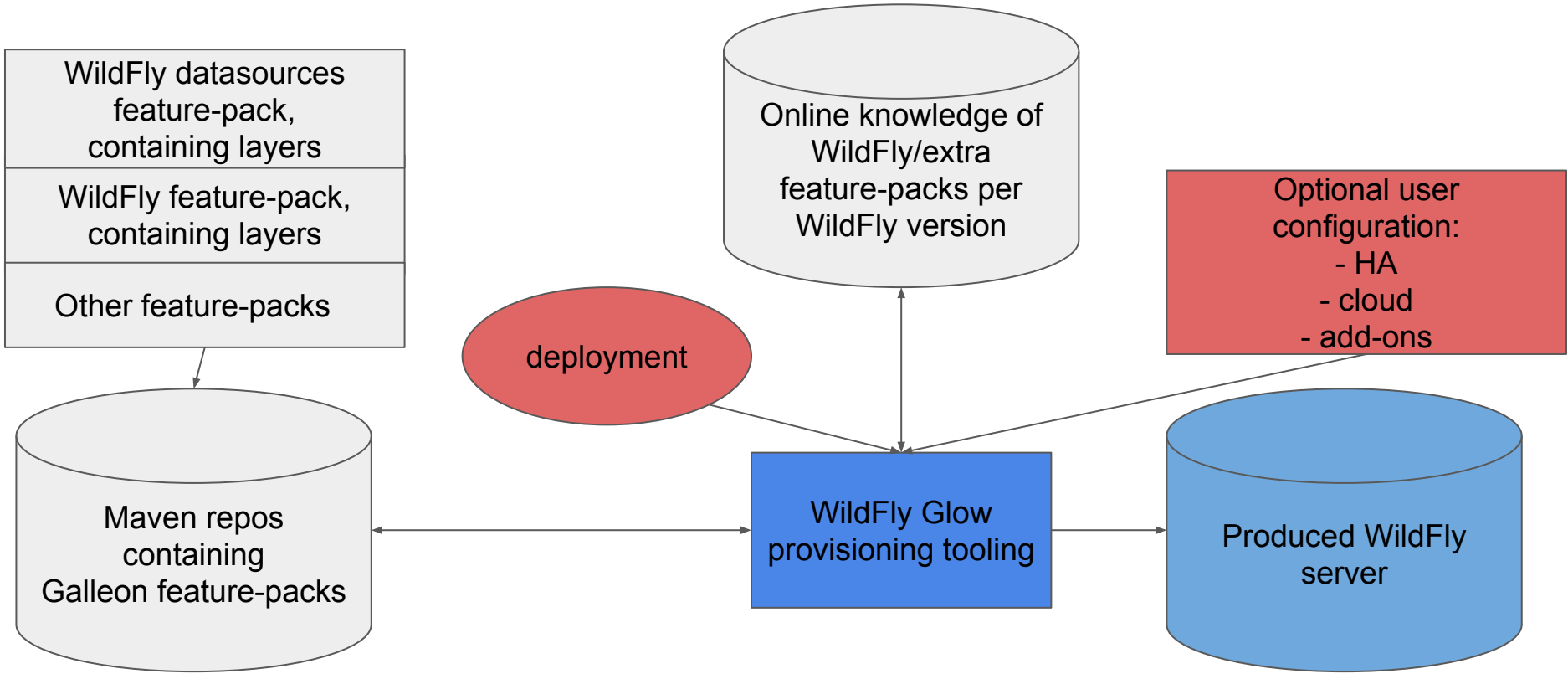
WildFly Glow Features (2/2)

- Handling of High Availability
  - WildFly Glow allows you to enable the “ha” profile to produce an High Available WildFly server
- Handling of 2 execution contexts
  - bare-metal (the default)
  - cloud, to execute on Openshift and/or Kubernetes.
- Handling of datasources
- A centralized knowledge of extra Galleon feature-packs compatible with WildFly and WildFly Preview. Currently:
  - Cloud, datasources, Keycloak SAML, GRPc, MyFaces, Microprofile-GraphQL, Resteasy Spring

WildFly Glow tooling

- **WildFly Glow CLI**, a standalone tool to scan your deployment(s) to produce a Galleon configuration, a WildFly server, a WildFly Bootable JAR or a docker image (direct deployment to OpenShift in progress).
- **Integration in WildFly Maven plugin**, no more explicit feature-packs and layers in the plugin configuration
- **WildFly Glow Arquillian Maven plugin** to scan your tests to produce WildFly server required to execute your tests

WildFly Glow workflow



How does it work?

- Leverage Galleon provisioning artifacts (Feature-packs and Layers)
- Relies on rules included in each Galleon Layer
  - Rules captures the content expected inside the deployment for the layer to be required
  - Rules express the High Availability capability of a layer
- Introduce the notion of add-on to extend discovered layers with layers that make sense according to the discovered ones
  - SSL, embedded/remote JMS brokers, postgresql/mysql/... datasources
  - WildFly CLI (jboss-cli, add-users, elytron tooling, .,..)
- Knows about High Availability, will automatically include HA Galleon layers
- Include built-in knowledge to identify missing datasources
- Relies on Jakarta EE core profile as the minimal server to enrich.

Galleon Layers rules

- Implementation detail known by WildFly Glow
- If you develop WildFly feature-packs, have a look to their documentation
- Metadata added to layers definitions
- Associate to a layer the:
  - API usage
  - Deployment descriptors/files content

A registry of feature-packs

- WildFly server extra features should be packaged as Galleon feature-pack
- Then registered in the registry.
- Open to contribute feature-packs that would bring added value to WildFly
- Currently:
  - Keycloak SAML
  - Grpc
  - MyFaces
  - Datasources
  - Cloud
  - Microprofile graphql
  -

WildFly Glow CLI Demos

- Local (download wildfly-glow from its releases page)
  - `./wildfly-glow --help`
  - `./wildfly-glow scan examples/kitchensink.war`
  - `./wildfly-glow scan examples/kitchensink.war --ha`
  - `./wildfly-glow scan examples/kitchensink.war --provision BOOTABLE_JAR`
  -
- Cloud, Openshift sandbox
  - Use of cloud option to fine tune the server configuration + enable health checks.
  - `./wildfly-glow scan examples/kitchensink.war --cloud`
  - `./wildfly-glow scan examples/kitchensink.war --cloud --provision DOCKER_IMAGE`
  - `sh ./openshift/push-image.sh`
  - `helm install kitchensink -f ./openshift/ helm.yaml wildfly/wildfly`

Numbers and limitations

- We have observed a reduction of 5% to 55% for disk usage and 5% to 32% for memory consumption with WildFly Glow compared to Galleon (based on WildFly quickstarts).
- Interesting simple project that compares zipped distribution, vs Galleon vs WildFly Glow.
- Limitations
  - We can’t discover layers when:
    - Java Reflection is used.
    - JNDI lookup is used. But we detect that JNDI API is used, advertise the usage points and allow for explicit addition of layers.

WildFly Maven plugin example (4.2.x)


```
…
<feature-packs>
  <feature-pack>
<location>org.wildfly:wildfly-galleon-pack:31.0.0.Final</location>
  </feature-pack>
</feature-packs>
<layers>
    <layer>ee-core-profile-server</layer>
    <layer>jaxrs</layer>
    <layer>ejb</layer>
    <layer>ejb-dist-cache</layer>
    <layer>jpa-distributed</layer>
</layers>
<excludedLayers>
    <layer>ejb-local-cache</layer>
</excludedLayers>
…
```

Same example, WildFly Maven plugin (5.x)


```
…
<discover-provisioning-info>
  <profile>ha</profile>
</discover-provisioning-info>
…
```

You can find examples in the [WildFly Quickstart Glow Preview branch]

Datasource support

- WildFly Glow detects that your deployment uses datasources
- It will suggest the set of known add-ons allowing to connect to database
- During second execution, the set of env variables to use to configure the datasource are displayed
- WildFly Glow prints the pieces found in your deployment (e.g.: JNDI name of the datasource)
- When starting the server you must set the env variables that WildFly Glow advertised

WildFly Glow CLI Database Demo

- `docker run --rm -p 5432:5432 -e POSTGRES_PASSWORD=frdemo -e POSTGRES_USER=frdemo postgres`
- `./wildfly-glow scan examples/todo-backend.war`
- `./wildfly-glow scan examples/todo-backend.war --add-ons=postgresql`
- `./wildfly-glow scan examples/todo-backend.war --add-ons=postgresql --provision=SERVER`
- `POSTGRESQL_DATABASE=frdemo POSTGRESQL_USER=frdemo POSTGRESQL_PASSWORD=frdemo POSTGRESQL_JNDI=java:jboss/datasources/ToDos sh server-31.0.1.Final/bin/standalone.sh &`
- `curl -X POST -H "Content-Type: application/json"  -d '{"title": "WildFly Mini Conference, March 2024!"}' http://localhost:8080/todo-backend`
- `curl http://127.0.0.1:8080/todo-backend`

Messaging add-ons

- WildFly Glow can identify that Messaging is required.
- Will advise the usage of an embedded Broker or (disjonctif) a remote Broker.

WildFly Glow CLI, Messaging demo

- `docker run --rm --name artemis -e AMQ_USER=admin -e AMQ_PASSWORD=admin -p8161:8161 -p61616:61616 -e AMQ_DATA_DIR=/home/jboss/data quay.io/artemiscloud/activemq-artemis-broker-kubernetes`
- `./wildfly-glow scan examples/remote-helloworld-mdb.war`
- `./wildfly-glow scan examples/remote-helloworld-mdb.war --add-ons=remote-activemq`
- `./wildfly-glow scan examples/remote-helloworld-mdb.war --add-ons=remote-activemq --provision=SERVER`
- `sh server-31.0.1.Final/bin/standalone.sh &`
- `curl http://localhost:8080/remote-helloworld-mdb/HelloWorldMDBServletClient`

WildFly quickstarts migrated to use Glow

- All applications used in these demos are from WildFly quickstarts
- They have been ported to use WildFly Glow
- Currently a preview branch
- 100% of quickstarts migrated
- Best source of information to help you start with WildFly Glow integration in Maven build

Native deployment to OpenShift

- That is a work in progress specified by this Issue.
- Current solution
  - Relies on locally built Docker image
  - Needs to push the image to the OpenShift cluster
  - Require that you set env variables to bind deployment to third parties (e.g.: PostgreSQL)
- Native OpenShift Support
  - Introduce a new type of provisioning: `OPENSHIFT`
  - No need for Docker, automated provisioning and deployment in OpenShift cluster
  - Handle third parties deployments (PostgreSQL DB, Keycloak server, Artemis JMS Broker...)
- Well suited for OpenShift testing/investigations
- Stay tuned, will be released in next Beta (very soon)

Status

- Beta level for WildFly 31
  - WildFly Glow is currently 1.0.0.Beta9
  - WildFly Maven Plugin 5.0.0.Beta3
  - All WildFly quickstarts ported to use WildFly Glow in preview branch
  - WildFly 31 testsuite has been ported to use WildFly Glow where applicable
  - WildFly Galleon feature-packs registry open to contributions
- Final level expected for WildFly 32
  - Final provisioning tooling
  - WildFly Quickstarts migrated to WildFly Glow and latest provisioning tooling

Resources

- WildFly Glow
  - Project: https://github.com/wildfly/wildfly-glow
  - Online documentation: http://docs.wildfly.org/wildfly-glow
- Recent blog posts:
  - Introduction
  - Vlog
  - Testing
  - Master the Boss article
- WildFly Galleon feature-packs registry
  - Project: https://github.com/wildfly/wildfly-galleon-feature-packs
  - Online documentation: http://docs.wildfly.org/wildfly-galleon-feature-packs
- WildFly Maven Plugin
  - Project: https://github.com/wildfly/wildfly-maven-plugin
  - Online documentation: https://docs.wildfly.org/wildfly-maven-plugin/releases/5.0/
- WildFly quickstarts
  - Migration to WildFly Glow: https://github.com/wildfly/quickstart/tree/glow-preview
- WildFly layers rules examples:
  - jaxrs: https://github.com/wildfly/wildfly/blob/main/ee-feature-pack/galleon-shared/src/main/resources/layers/standalone/jaxrs/layer-spec.xml
  - ejb: https://github.com/wildfly/wildfly/blob/main/ee-feature-pack/galleon-shared/src/main/resources/layers/standalone/ejb/layer-spec.xml

THANK-YOU!

Q&A

Feedback form: https://tinyurl.com/wildfly