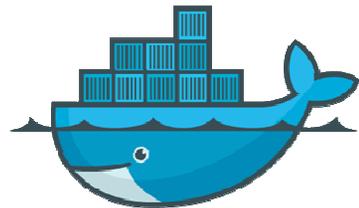


Deployment Patterns using Docker and Chef

Sandeep Chellingi

Sandeep.chellingi@prolifics.com

Agenda



+



+

urban{code}

Deploy

docker

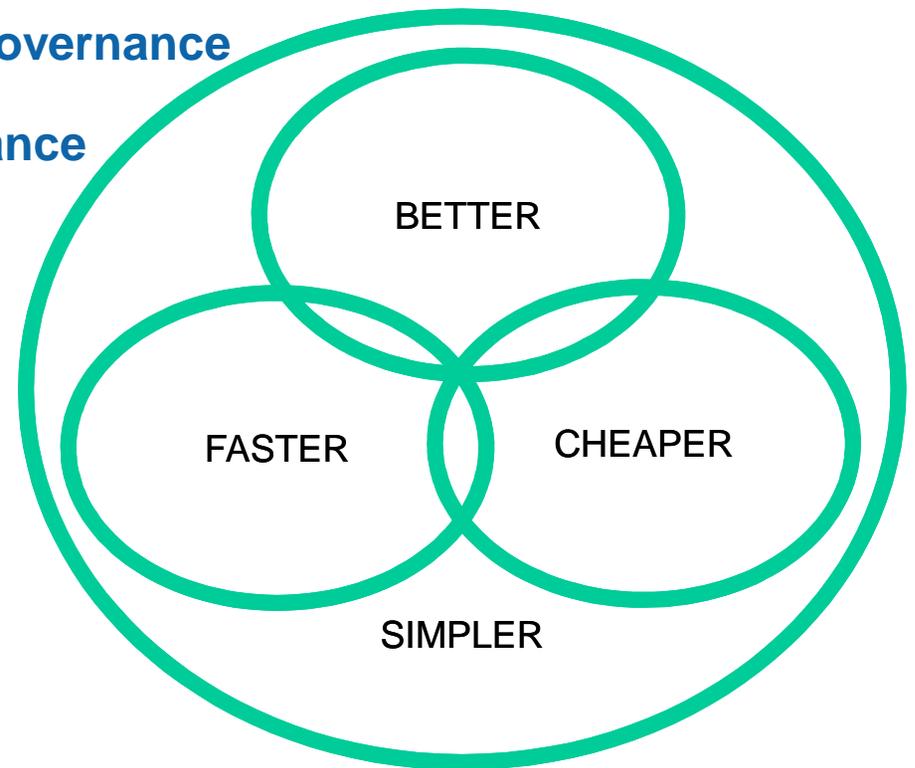
CHEF™

Rapid Provisioning + Automated and Managed Deployment

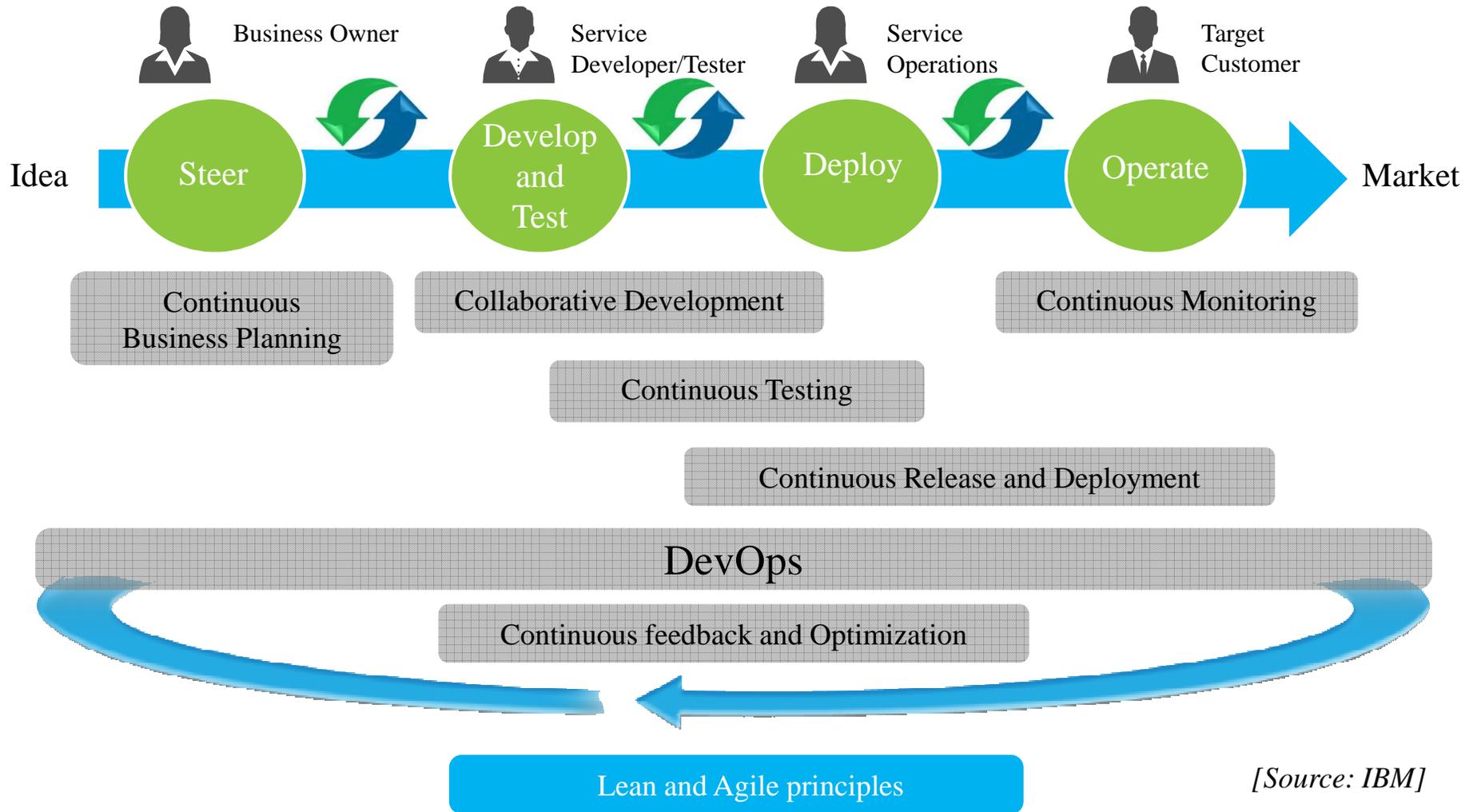
- **IT Challenges - Use-cases**
- **What is Docker ?**
- **What is Chef ?**
- **What is IBM Urban Code Deploy ?**
- **Demo**
- **Questions & Answers**

IT challenges

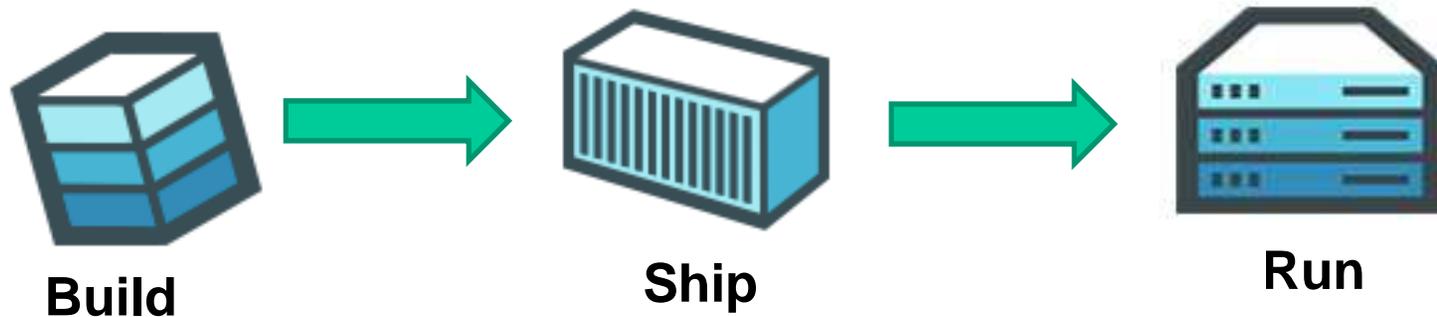
- Speed and complexity
- Security and auditability
- Enforcement of standards and governance
- Regulatory controls and compliance
- Consistency and repeatability
- Resource constraints
- Operational cost reduction



DevOps Overview



What is Docker?



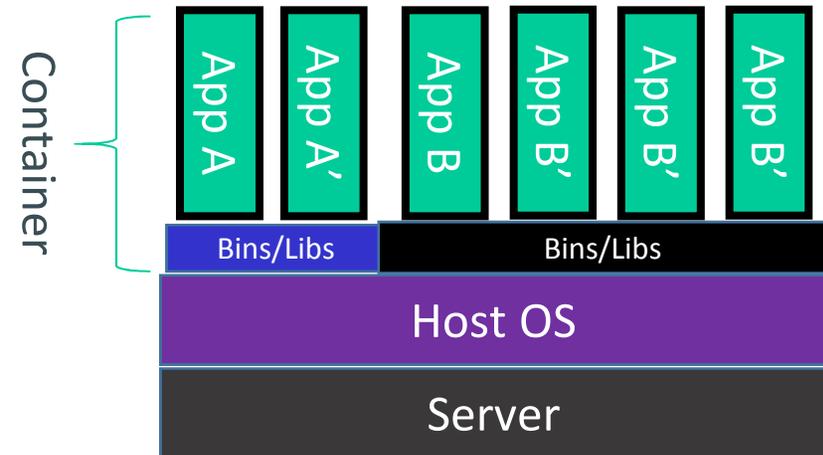
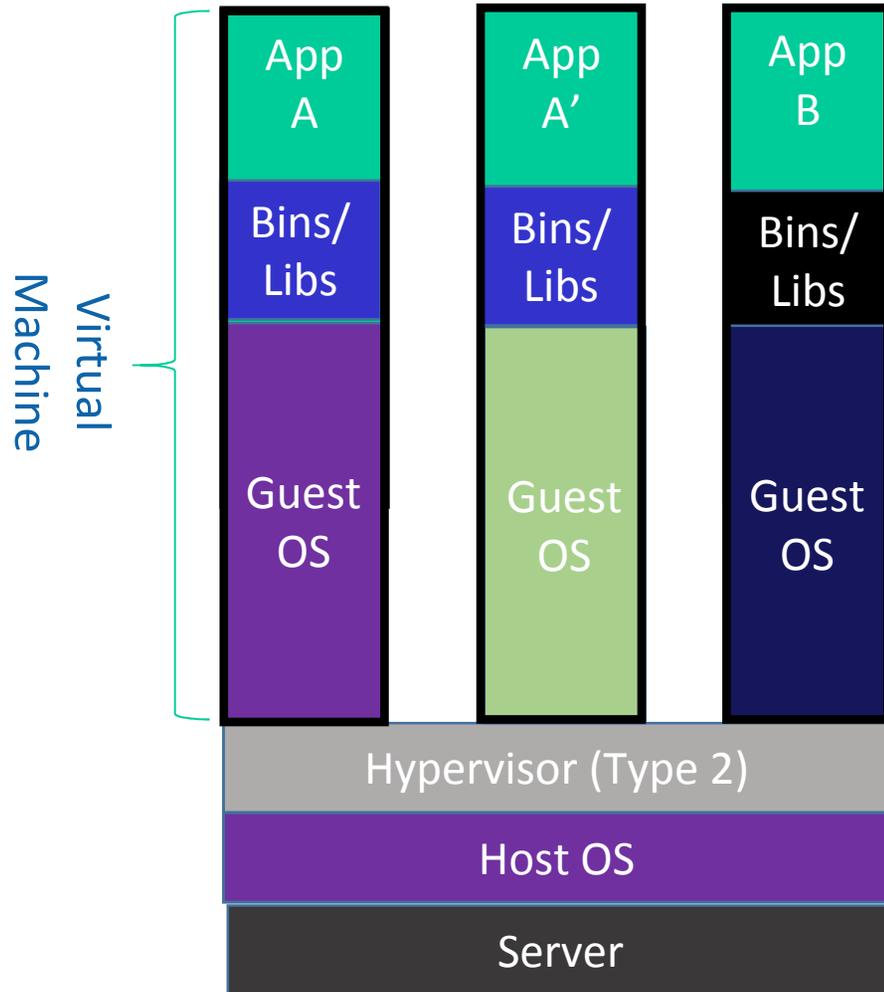
Docker build, ship, run any app, anywhere.

“**Docker** is an open-source project to easily create lightweight, portable, self-sufficient containers from any application. The same container that a developer builds and tests on a laptop can run at scale, in production, on VMs, bare metal, OpenStack clusters, public clouds and more.”

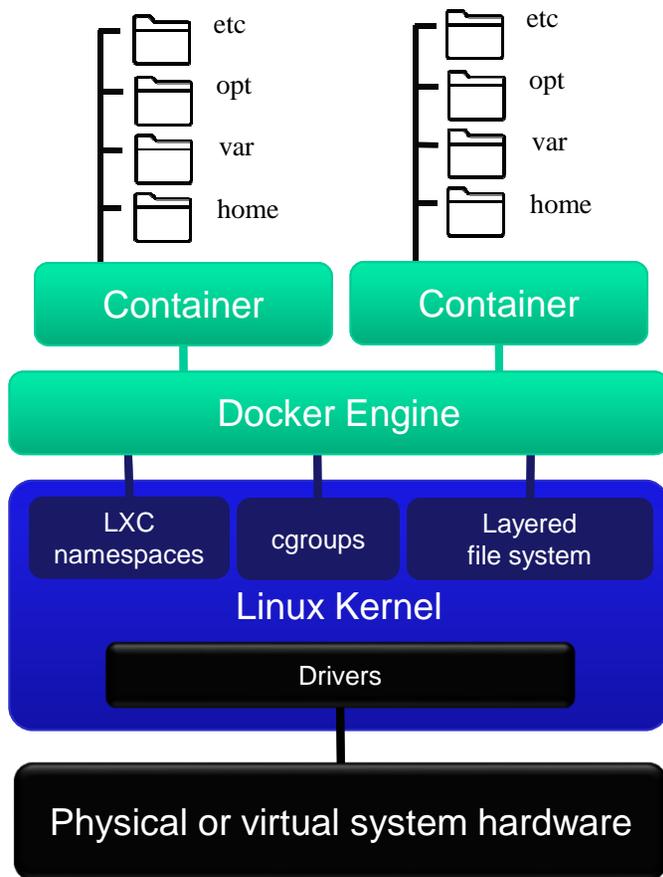
What is a Container ?

Containers:

- Isolated name space (processes partition)
- Shared system resources, but run-time isolation of application resources and data.
- Network and volume mappers.



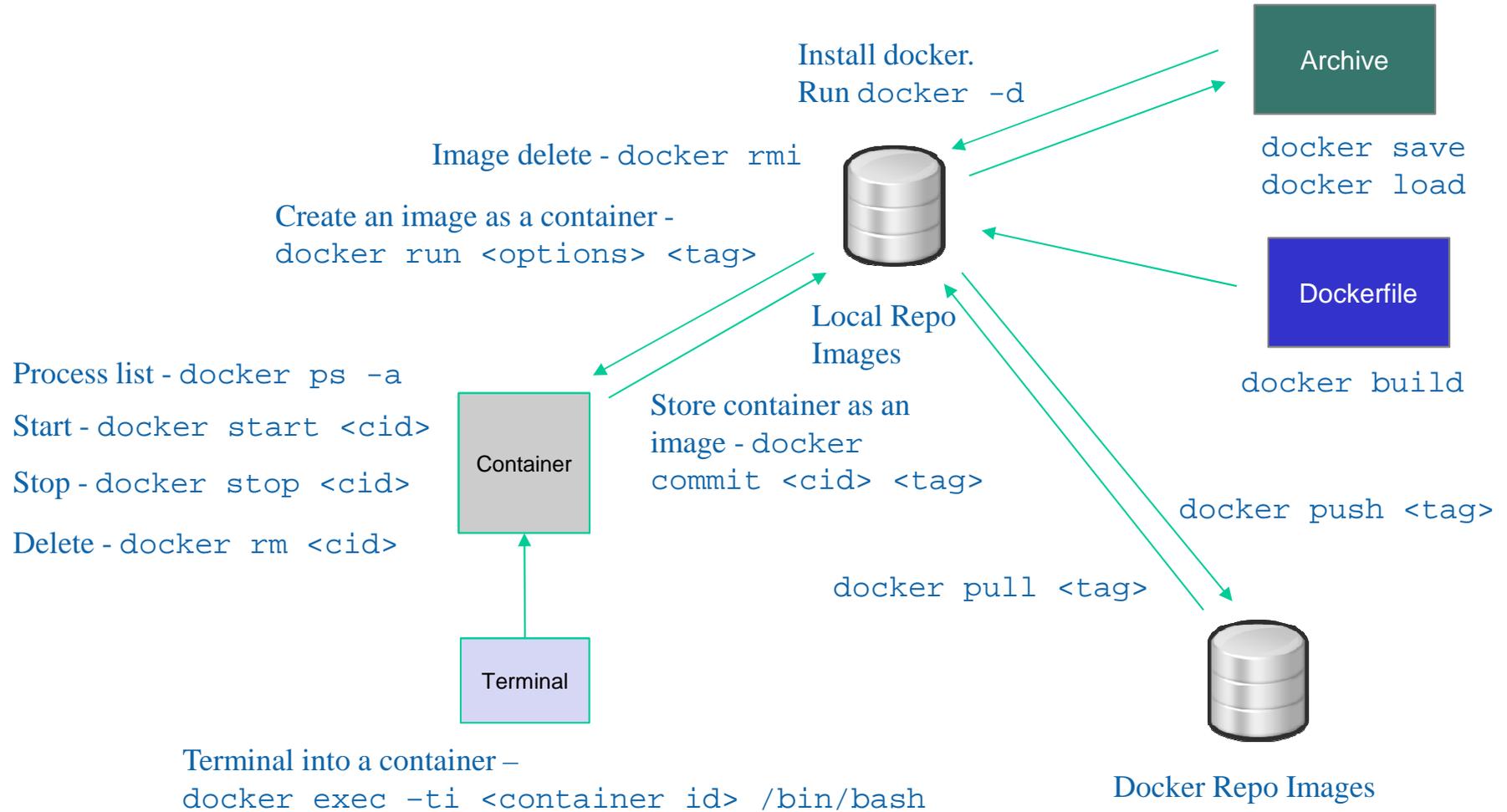
Container architecture and value



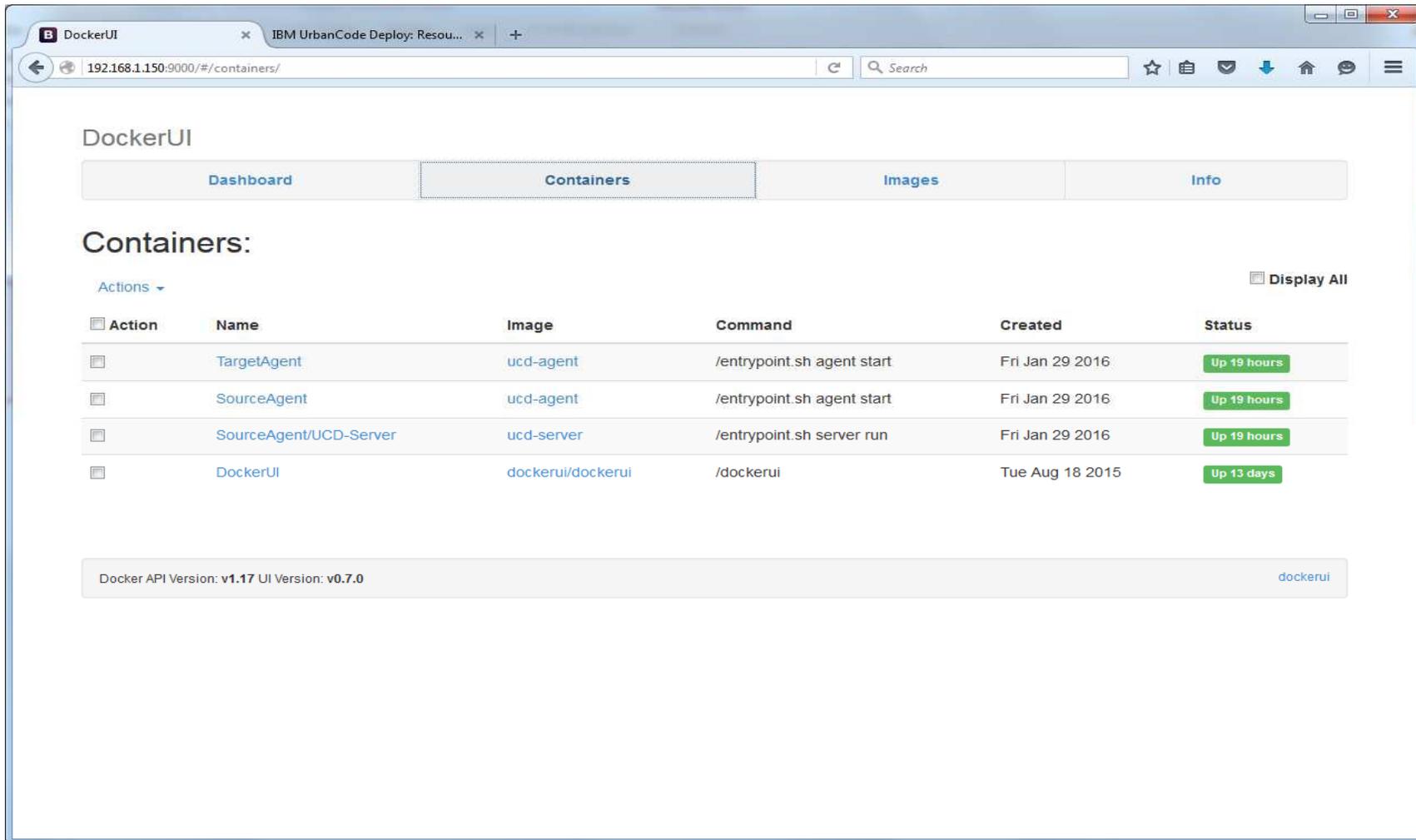
If you're focusing on the architecture **you might be missing the point...**

- We are using **already allocated resources**, so start up and shut down is **quick**.
- We are using a **layered file system** so we only need to store or retrieve deltas, which **saves network bandwidth**.
- The container runs on any X86 system with a Docker Engine, so **consistency and reuse is extremely high**.

Docker Fundamentals



Docker UI



The screenshot shows the DockerUI web interface in a browser window. The browser tabs include 'DockerUI' and 'IBM UrbanCode Deploy: Resou...'. The address bar shows '192.168.1.150:9000/#/containers/'. The interface has a navigation bar with 'Dashboard', 'Containers', 'Images', and 'Info'. The 'Containers' section is active, displaying a table of containers. The table has columns for Action, Name, Image, Command, Created, and Status. There are four containers listed: TargetAgent, SourceAgent, SourceAgent/UCD-Server, and DockerUI. Each container has a status indicator showing it is 'Up' for a certain duration. At the bottom, there is a footer showing 'Docker API Version: v1.17 UI Version: v0.7.0' and the name 'dockerui'.

DockerUI

Dashboard Containers Images Info

Containers:

Actions Display All

Action	Name	Image	Command	Created	Status
<input type="checkbox"/>	TargetAgent	ucd-agent	/entrypoint.sh agent start	Fri Jan 29 2016	Up 19 hours
<input type="checkbox"/>	SourceAgent	ucd-agent	/entrypoint.sh agent start	Fri Jan 29 2016	Up 19 hours
<input type="checkbox"/>	SourceAgent/UCD-Server	ucd-server	/entrypoint.sh server run	Fri Jan 29 2016	Up 19 hours
<input type="checkbox"/>	DockerUI	dockerui/dockerui	/dockerui	Tue Aug 18 2015	Up 13 days

Docker API Version: v1.17 UI Version: v0.7.0 dockerui

What is Chef ?



Chef is an open source software agent that automates your infrastructure by turning it into code.

Chef handled infrastructure becomes dynamic, versionable, human-readable, and testable.

Chef automates how infrastructure is configured, deployed, and managed across your network, no matter its size.

Chef Leverages to easily and quickly provision, manage, and adapt infrastructure in response to always changing needs.

What is Chef ?

Chef supports management of servers infrastructure in the cloud, on-premises, or in a hybrid environment.

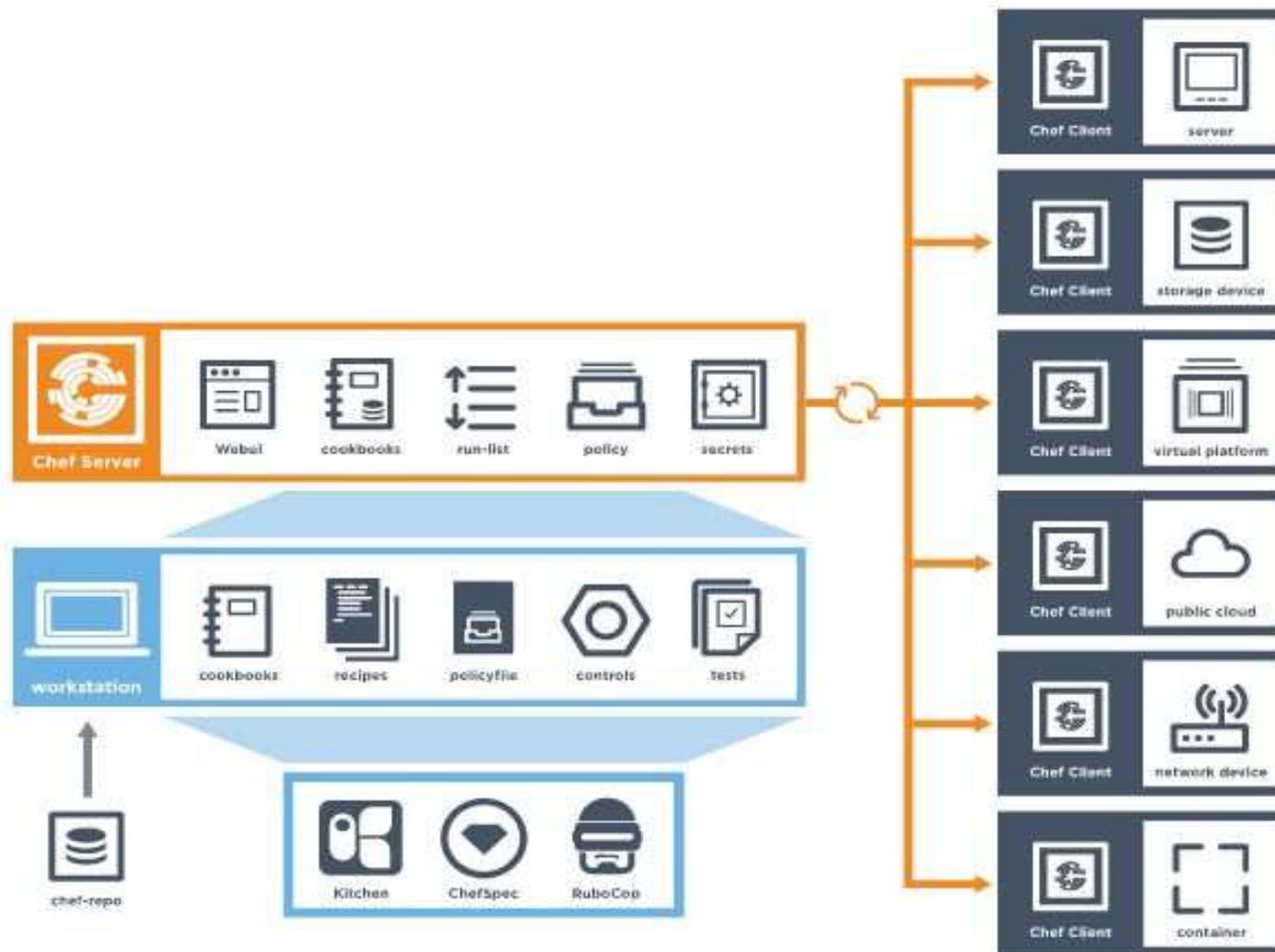
Nodes are the machines – physical , virtual, cloud, and so on that are under management by Chef.

The chef-client is installed on each node and client performs the automation on the nodes.

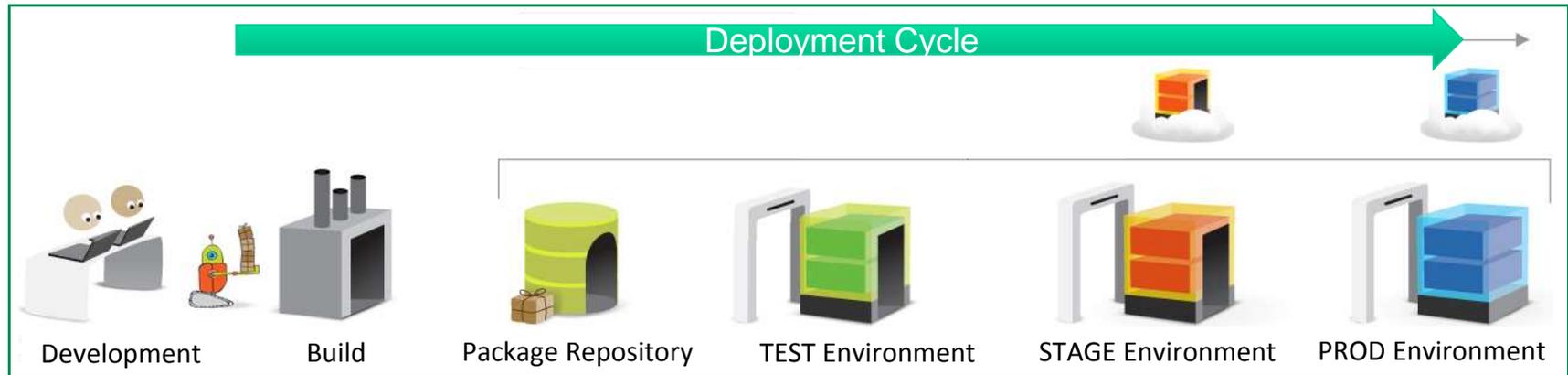
Chef server is the backbone to create and manage flexible, dynamic infrastructure (chef) nodes, across datacenters, public and private clouds , and in heterogeneous environments.

Chef Server acts as a configuration data hub, storing cookbooks and policies.

What is Chef ?

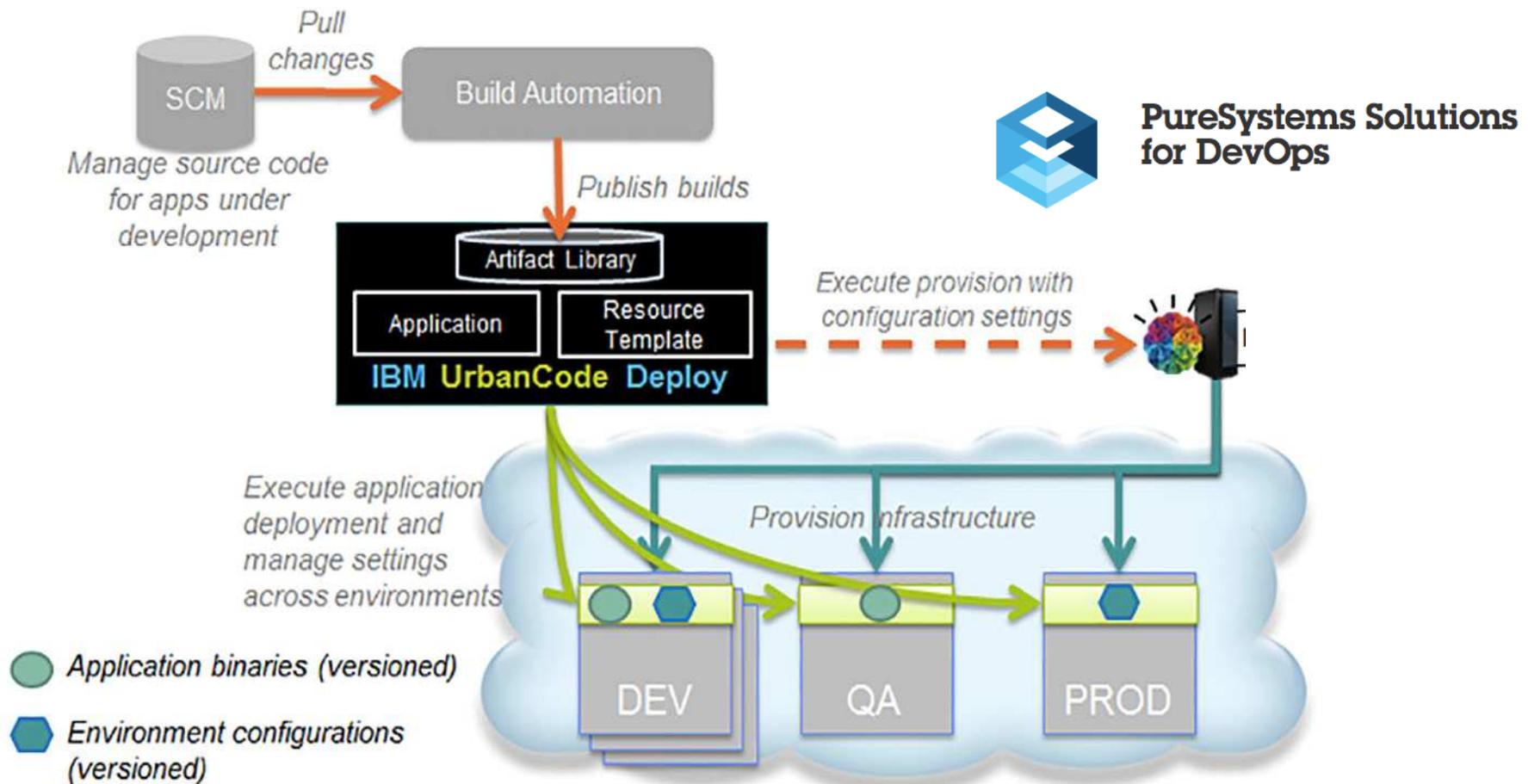


What is UrbanCode Deploy ?

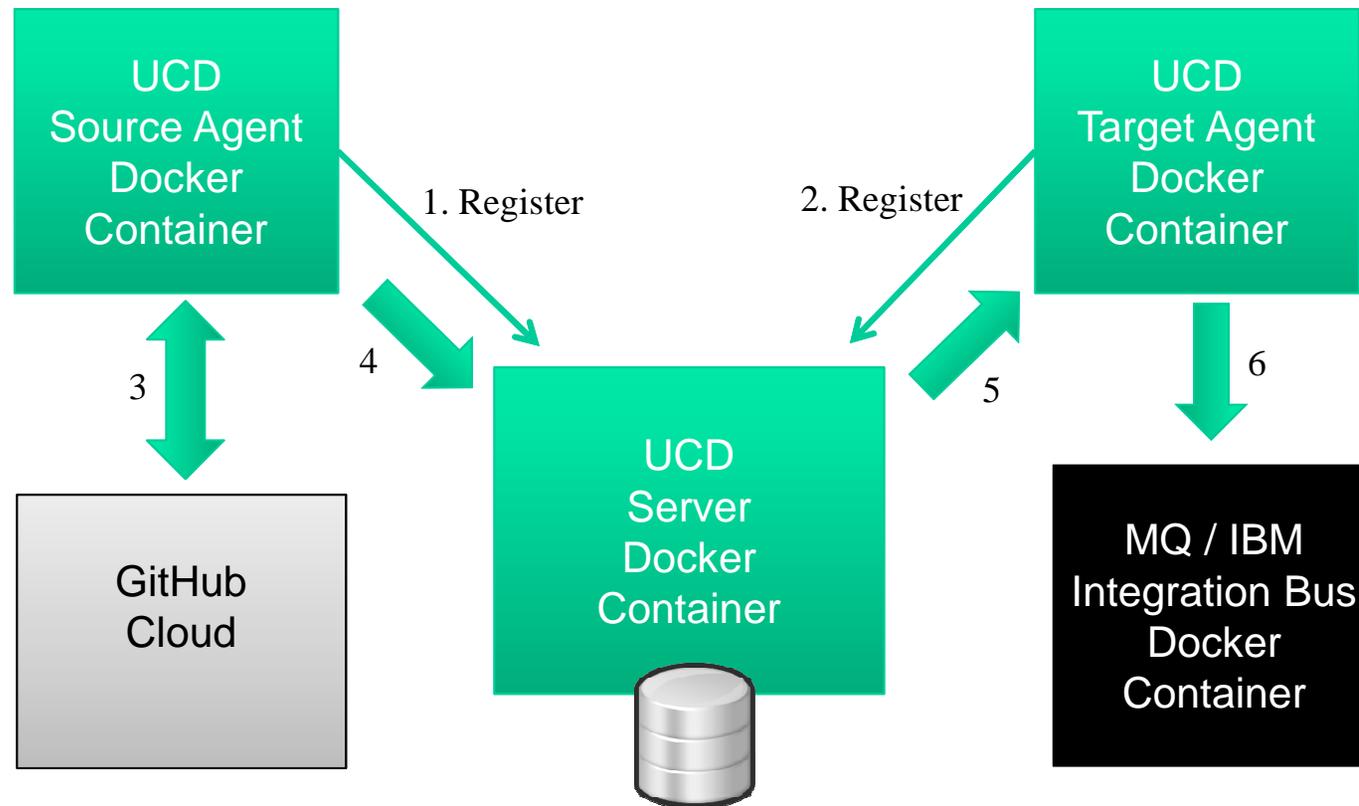


- Deployment automation
- Application and environment visibility and transparency
- Process flexibility
- Access controls and audit controls

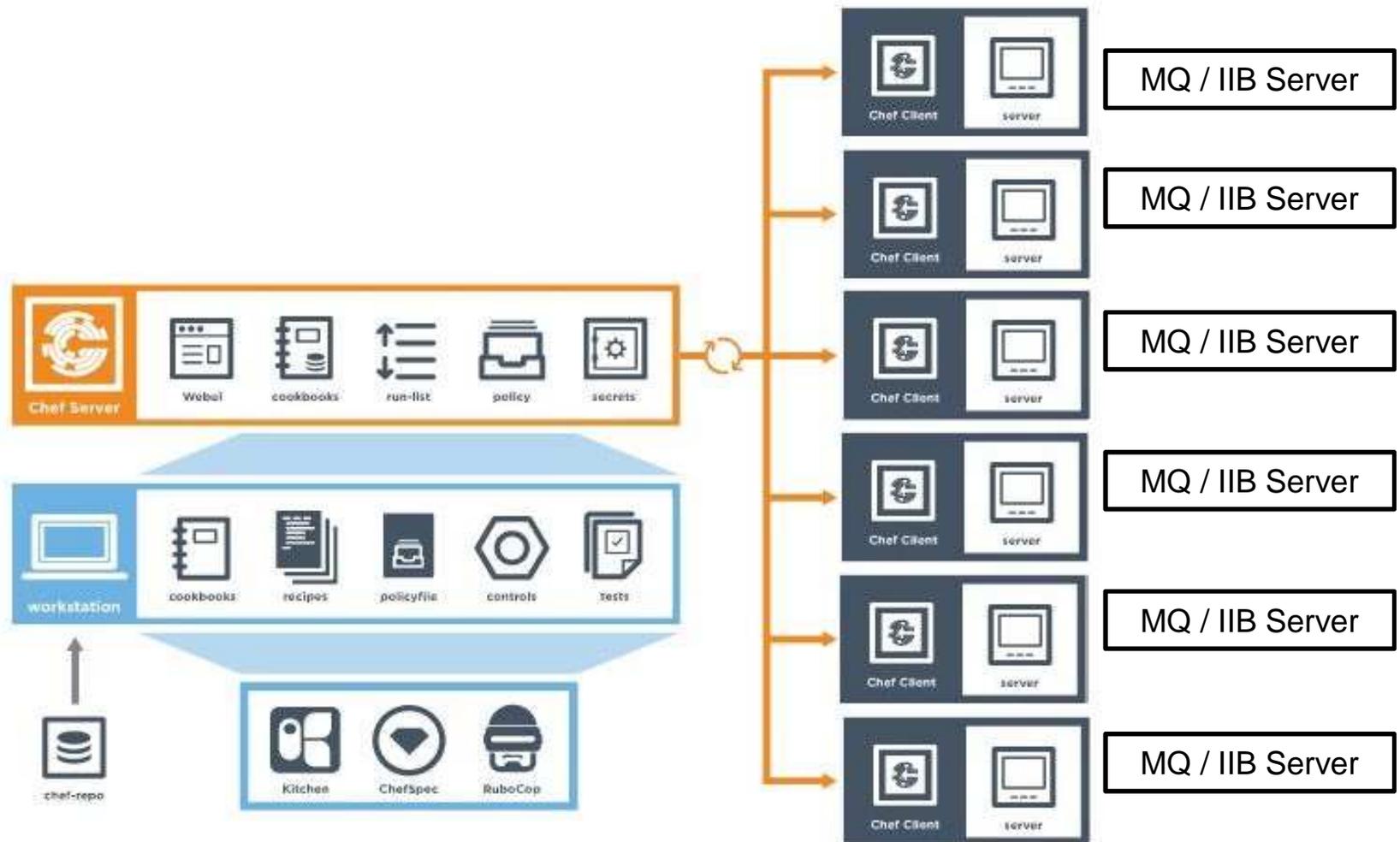
What is UrbanCode Deploy ?



UrbanCode Deploy – IIB code deploy



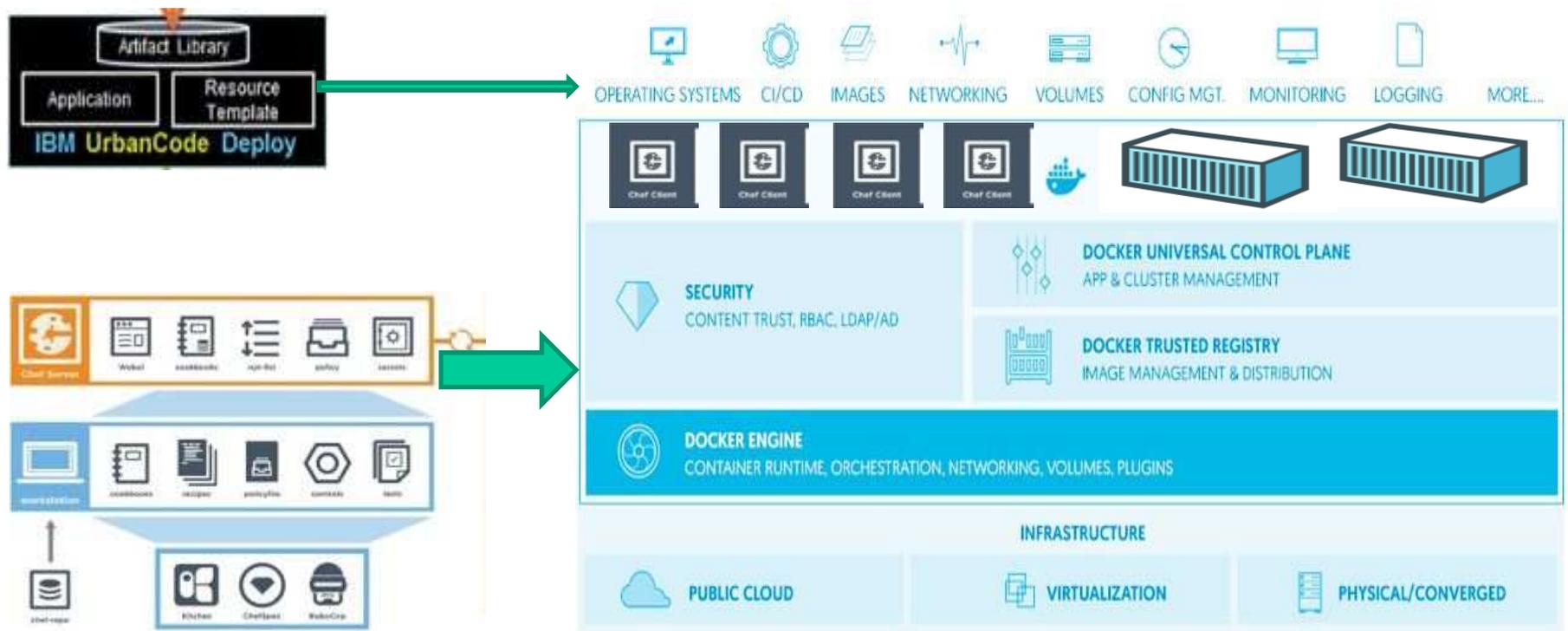
Chef – MQ & IIB Deploy Pattern



Docker CAAS – Deploy Pattern



Chef + Docker + UCD – Deploy Pattern



Demo

Demo – Docker Chef MQ – IIB

```
root@localhost:/home/vagrant
[root@localhost vagrant]# docker load -i /vagrant/iibmqnode.tar.gz
[root@localhost vagrant]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
iibmqnode            latest             a8439b089193      5 weeks ago       2.974 GB
[root@localhost vagrant]#

root@localhost:/home/vagrant
[root@localhost vagrant]# docker run -d -v /HA:/HA --name node1 iibmqnode tail -f /dev/null
F83657614cc8d67ce8ad080ea125f32de0c00a5d7fba867d3091bed9d12e275f
[root@localhost vagrant]# docker exec -it node1 /bin/bash
[root@F83657614cc8 /]#

root@localhost:/home/vagrant
[root@F83657614cc8 /]# cd /HA/chef-repo
[root@F83657614cc8 chef-repo]# chef-client -z -r 'recipe[iib::create_broker_w_mq_ha]'
[2016-03-29T20:25:10+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.4.1
resolving cookbooks for run list: ['iib::create_broker_w_mq_ha']
Synchronizing Cookbooks:
- iib
- mqv8
Compiling Cookbooks...
Converging 7 resources
Recipe: iib::create_broker_w_mq_ha
* bash[configure_groups] action run
  - execute "bash" "/tmp/chef-script20160329-22-ucqxt0"
* template[/tmp/iib_mqsc.in] action create
  - create new file /tmp/iib_mqsc.in
  - update content in file /tmp/iib_mqsc.in from none to db122d
  --- /tmp/iib_mqsc.in          2016-03-29 20:25:11.752495348 +0000
  +++ /tmp/chef-rendered-template20160329-22-gablur  2016-03-29 20:25:11.752495348 +0000
  @@ -1,3 @@
  +DEF QL(IIB_QM_HA.DLQ) REPLACE
  +
  - change owner from '' to 'mqbrkusr'
* bash[create_dirs] action run
  - execute "bash" "/tmp/chef-script20160329-22-171w5v4"
* bash[create_iib_qm] action run
  - execute "bash" "/tmp/chef-script20160329-22-1hjkhd"
* bash[create_broker] action run
  - execute "bash" "/tmp/chef-script20160329-22-4mlg84"
* bash[create_queues] action run
  - execute "bash" "/tmp/chef-script20160329-22-ta9uxk"
* bash[create_broker_server] action run
  - execute "bash" "/tmp/chef-script20160329-22-13byt0u"

Running handlers:
Running handlers complete
Chef Client finished, 7/7 resources updated in 58.296155021 seconds
[root@F83657614cc8 chef-repo]#
```

Demo – Docker Chef MQ – IIB

```
root@localhost: /home/vagrant
[vagrant@localhost ~]$ sudo su
[root@localhost vagrant]# docker run -d -v /HA:/HA --name node2 iibmqnode tail -f /dev/null
1567d905d55d4b7e12df2815e7cb227bfa7a16ba174e1830ba044b8c41a99d4f
[root@localhost vagrant]# docker exec -it node2 /bin/bash
[root@1567d905d55d /]# cd /HA/chef-repo && chef-client -z -r 'recipe[iib::create_broker_w_mq_ha]'
[2016-03-29T20:36:00+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.4.1
resolving cookbooks for run list: ["iib::create_broker_w_mq_ha"]
Synchronizing Cookbooks:
- iib
- mqv8
Compiling Cookbooks...
Converging 7 resources
Recipe: iib::create_broker_w_mq_ha
 * bash[configure_groups] action run
   - execute "bash" "/tmp/chef-script20160329-22-1v54uut"
 * template[/tmp/iib_mqsc.in] action create
   - create new file /tmp/iib_mqsc.in
   - update content in file /tmp/iib_mqsc.in from none to db122d
--- /tmp/iib_mqsc.in          2016-03-29 20:36:01.890385118 +0000
+++ /tmp/chef-rendered-template20160329-22-1cfdweu 2016-03-29 20:36:01.890385118 +0000
@@ -1,3 @@
+DEF QL(IIB_QM_HA.DLQ) REPLACE
+
- change owner from '' to 'mqbrksr'
 * bash[create_dirs] action run (skipped due to not_if)
 * bash[create_iib_qm] action run
   - execute "bash" "/tmp/chef-script20160329-22-74o6tt"
 * bash[create_broker] action run
   - execute "bash" "/tmp/chef-script20160329-22-1o1h1pp"
 * bash[create_queues] action run (skipped due to not_if)
 * bash[create_broker_server] action run (skipped due to only_if)

Running handlers:
Running handlers complete
Chef Client finished, 4/4 resources updated in 12.965986025 seconds
[root@1567d905d55d chef-repo]#
```

```
root@localhost: /home/vagrant
[root@f83657614cc8 chef-repo]# netstat -an | grep 4414
tcp        0      0 :::4414          :::*              LISTEN
[root@f83657614cc8 chef-repo]#
```

Demo – Docker Chef MQ – IIB

```
root@localhost:/home/vagrant
[root@1567d905d55d chef-repo]# source /opt/IBM/iib-10.0.0.1/server/bin/mqsiprofile

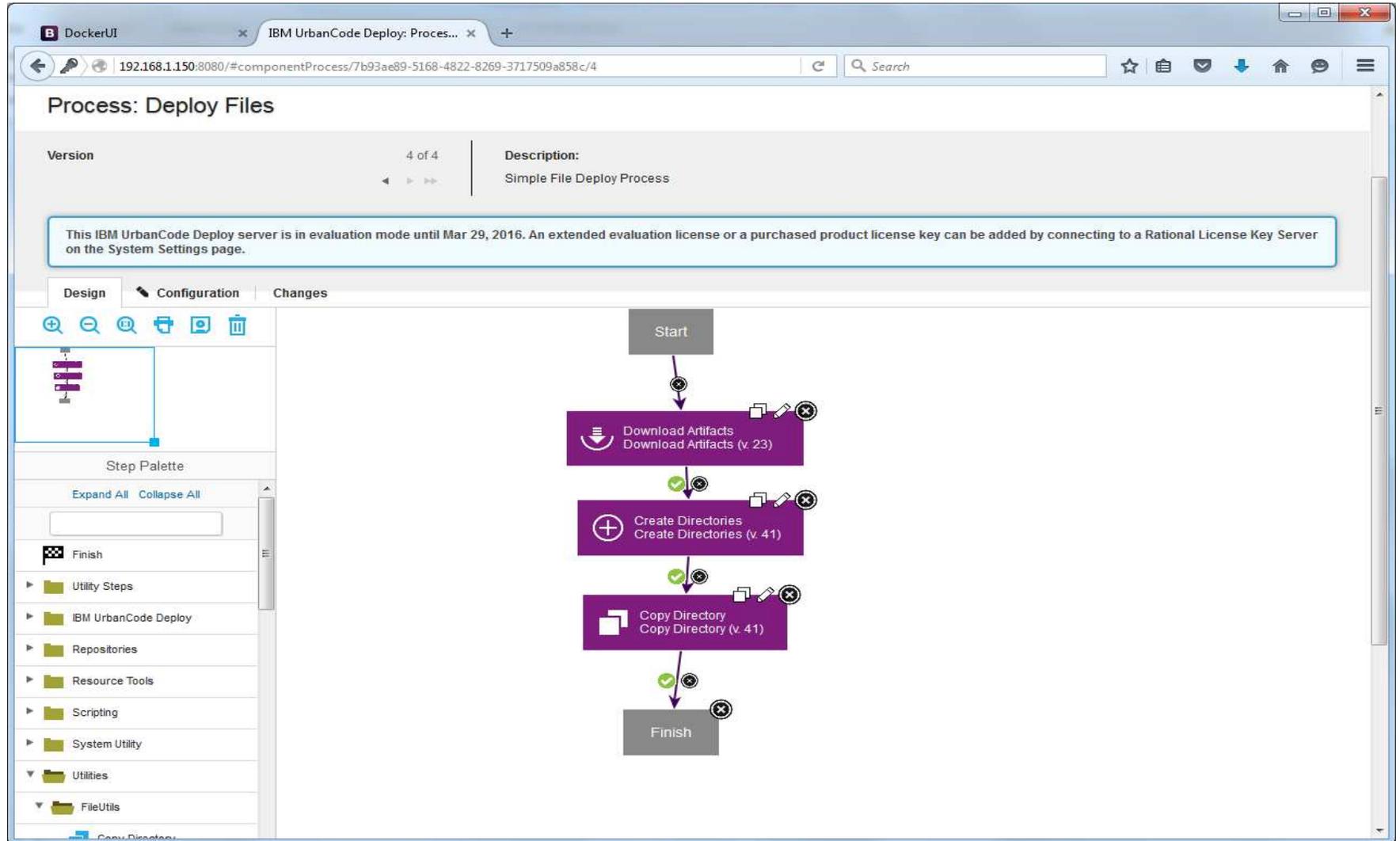
MQSI 10.0.0.1
/opt/IBM/iib-10.0.0.1/server

[root@1567d905d55d chef-repo]# mqsilist
BIP1296I: Integration node 'IBNODE_HA' is stopped. It is a multi-instance integration node and will be started as a WebSphere MQ service by queue manager 'IIB_QM_HA'.
BIP8071I: Successful command completion.
[root@1567d905d55d chef-repo]#
```

```
root@localhost:/home/vagrant
[root@f83657614cc8 chef-repo]#
[root@f83657614cc8 chef-repo]# su mqm -c "/opt/mqm/bin/endmqm -s IIB_QM_HA"
Quiesce request accepted. The queue manager will stop when all outstanding work is complete, permitting switchover to a standby instance.
[root@f83657614cc8 chef-repo]#
```

```
root@localhost:/home/vagrant
[root@1567d905d55d chef-repo]# netstat -an | grep 4414
tcp        0      0 :::4414          :::*              LISTEN
[root@1567d905d55d chef-repo]# mqsilist
BIP1295I: Integration node 'IBNODE_HA' is an active multi-instance or High Availability integration node that is running on queue manager 'IIB_QM_HA'.
BIP8071I: Successful command completion.
[root@1567d905d55d chef-repo]#
```

Demo – UCD - Process Designer



Demo- UCD - Application Environments

The screenshot displays the IBM UrbanCode Deploy web interface. The browser address bar shows the URL: 192.168.1.150:8080/#application/4f39dc23-93e4-4287-af06-73f49daf6069/environments. The page title is "Application: Target_Application". The breadcrumb navigation is "Home > Applications > Target_Application". The page shows the application was created by "admin" on "2/2/2016, 9:39 AM". A notification box states: "This IBM UrbanCode Deploy server is in evaluation mode until Mar 29, 2016. An extended evaluation license or a purchased product license key can be added by connecting to a Rational License Key Server on the System Settings page." The "Environments" tab is selected, showing a table of environments: DEV, QUA, and PRD. Each environment has a "Snapshot" of "None" and a "Compliancy" status. The DEV environment has a compliancy of 1/1, while QUA and PRD have 0/0. A "Create Environment" button is visible at the top left of the environment list.

Environment Name	Snapshot	Compliancy
DEV	None	1 / 1
QUA	None	0 / 0
PRD	None	0 / 0

Demo – UCD - Resource Assignments

The screenshot shows the IBM UrbanCode Deploy web interface. The browser address bar indicates the URL: 192.168.1.150:8080/#environment/018a3e47-c144-4691-ae0-8c9a7ff1bd86. The page title is "Environment: DEV for Target_Application". A notification box states: "This IBM UrbanCode Deploy server is in evaluation mode until Mar 29, 2016. An extended evaluation license or a purchased product license key can be added by connecting to a Rational License Key Server on the System Settings page." The navigation menu includes Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The "Resources" tab is active, showing a table of resource assignments. The table has columns for Name, Inventory, Status, and Description. A single resource is listed with an inventory of 1.0. Below the table are "Refresh" and "Print" links.

Home > Applications > Target_Application > Environments > Environment: DEV

Environment: DEV for Target_Application

This IBM UrbanCode Deploy server is in evaluation mode until Mar 29, 2016. An extended evaluation license or a purchased product license key can be added by connecting to a Rational License Key Server on the System Settings page.

Resources | History | Calendar | Configuration | Changes

Compliant: 1/1

Add Base Resources | Select All... | Actions... | Show | Expand All | Collapse All

<input type="checkbox"/>	Name	Inventory	Status	Description
<input type="checkbox"/>	Resource Name Tags			
<input type="checkbox"/>	Target_Resource_Group / Target_Agent_01 (View Agent) / File_Target / Component_File_Transfer_Example	1.0		

Refresh Print

Demo – UCD - Deploy

The screenshot displays the IBM UrbanCode Deploy web interface. The browser address bar shows the URL `192.168.1.150:8080/#application/4f39dc23-93e4-4287-af06-73f49daf6069`. The application title is "IBM UrbanCode Deploy" and the user is logged in as "admin". The navigation menu includes Dashboard, Components, Applications, Configuration, Processes, Resources, Calendar, Work Items, Reports, and Settings. The current page is "Application: Target_Application".

A modal dialog box titled "Run Process on DEV" is open in the center. It contains the following fields and options:

- Only Changed Versions:**
- Process*:** A dropdown menu with "Deploy Target Application" selected.
- Select a snapshot, or choose versions for individual components:** A dropdown menu for "Snapshot".
- Schedule Deployment?:**
- Description:** A text input field.

At the bottom of the dialog are "Submit" and "Cancel" buttons. In the background, the application details for "Target_Application" are visible, showing it was created by "admin" on "2/2/2016, 9:39 AM". There are three environments listed: DEV, QUA, and PRD, each with a play button icon and a "Compliance: 0 / 0" status.

Demo – UCD - Deployment in Progress

The screenshot displays the DockerUI interface for an IBM UrbanCode Deploy process. The browser address bar shows the URL: 192.168.1.150:8080/#applicationProcessRequest/329074e0-6953-4caf-ad09-3e4ee5497b21/log. The main heading is "Application Process Request: Target_Application".

Process Details:

- Process: Deploy Target Application (Version 3)
- Environment: DEV
- Only Changed Versions: false
- Date Requested: 2/2/2016, 2:49 PM
- Requested By: admin
- Scheduled For: 2/2/2016, 2:49 PM

Navigation buttons include "View Deployment Request" and "Process Request". A message box states: "This IBM UrbanCode Deploy server is in evaluation mode until Mar 29, 2016. An extended evaluation license or a purchased product license key can be added by connecting to a Rational License Key Server on the System Settings page."

Navigation tabs: Log, Properties, Manifest, Configuration Changes, Inventory Changes.

Execution Summary:

Buttons: Pause, Cancel, Download All Logs. Links: Expand All, Collapse All.

Step	Progress	Start Time	Duration	Status
1. Install Component-File-Example	0 / 1	2:49:30 PM	0:00:13	Running
Component_File_Transfer_Example	0 / 1	2:49:31 PM	0:00:12	Running
Deploy Files (Component_File_Example 1.0)	::	2:49:31 PM	0:00:12	Running
1. Download Artifacts	::	2:49:32 PM	0:00:11	Running
2. Create Directories				Not Started
3. Copy Directory				Not Started
Total Execution	0 / 1	2:49:30 PM	0:00:13	Running

Demo – UCD - Deployment Complete

The screenshot shows the DockerUI interface for an IBM UrbanCode Deploy process. The browser address bar indicates the URL: `192.168.1.150:8080/#applicationProcessRequest/329074e0-6953-4caf-ad09-3e4ee5497b21/log`. The main content area features a navigation bar with 'View Deployment Request' and 'Process Request' buttons. A message box states: 'This IBM UrbanCode Deploy server is in evaluation mode until Mar 29, 2016. An extended evaluation license or a purchased product license key can be added by connecting to a Rational License Key Server on the System Settings page.' Below this, there are tabs for 'Log', 'Properties', 'Manifest', 'Configuration Changes', and 'Inventory Changes'. The 'Execution' section shows a progress bar for the entire process, which is 1/1 complete and successful. A table below provides a detailed breakdown of the execution steps.

Step	Progress	Start Time	Duration	Status
1. Install Component-File-Example	1 / 1	2:49:30 PM	0:00:27	Success
Component_File_Transfer_Example	1 / 1	2:49:31 PM	0:00:26	Success
Deploy Files (Component_File_Example 1.0)	::	2:49:31 PM	0:00:26	Success
1. Download Artifacts	::	2:49:32 PM	0:00:14	Success
2. Create Directories	::	2:49:47 PM	0:00:05	Success
3. Copy Directory	::	2:49:52 PM	0:00:05	Success
Total Execution	1 / 1	2:49:30 PM	0:00:27	Success

Questions & Answers

