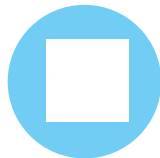




OneDevOps | Build-On

A Solution for Scalable on Demand CI Infrastructure

Setting up a CI Ecosystem – The Complications...



Infrastructure needs
differ with
Technologies /
applications



Multiple plug-ins
needed for different
Technologies /
applications



Multiple
Configurations are
needed with each
set-up



Time-consuming
Manual
Hardware scaling

The Voice of the industry...

<5%

Average Hardware
Utilization when using
Jenkins (Source: eBay)

>30 Days

Average Time taken to
set-up Jenkins Master-
Slave ecosystem

Setting up a CI Ecosystem – The Impact

Build Engineer



Limited
Scalability



Cumbersome
Maintenance



Error Prone
Provisioning

Developer



Lower
productivity



Snowflake build
configurations



Longer
Waiting times

Continuous Integration - Tools alone not enough!

Container Technology – Gaining momentum

200%

*Growth in usage of
Docker since 2015*

41%

*People who are already
using Containers in 2016*

18%

*Container usage that
has reached Production*

Source: Rightscale cloud report 2016

Benefits of Using Containers for CI

Consume lesser
Hardware resources

Can be provisioned on-
demand and can be freed
after execution

Self-contained capsules
that can be customized
for specific applications

Provide unlimited
scalability



Create containers On-demand with

OneDevOps | Build-On

Get **Unlimited scalability** for Continuous Integration
Increase utilization and agility of **Build infrastructure**

Build-On: Key Features

Optimized Build Infrastructure



- Self-Contained container provisioning
- Minimal manual intervention
- VMs and Containers freed after the execution, thus optimizing infrastructure

Code-Based Management of Environments & Pipeline



- Enables Developers to descriptively define pipeline & Jobs as simple YAML files
- Codified approach for standing up Jenkins and the associated Plugins ecosystem

Scalable, Self-Controlled Build Environments



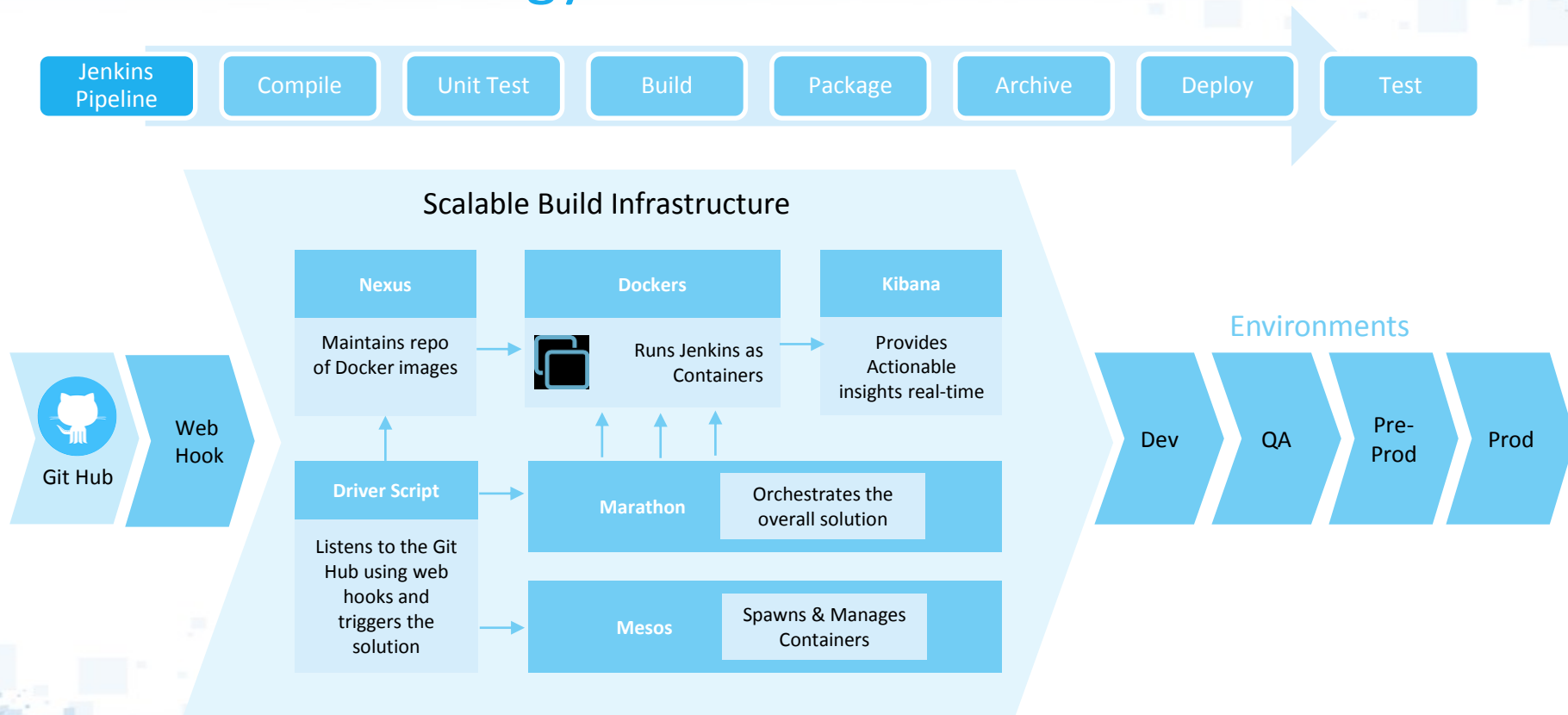
- Built based on the best names in Industry – Docker, Apache Mesos, Marathon and Jenkins
- Scale up and Scale Down vertically and horizontally based on need

Flexibility in Supporting Multiple Technologies



- Extensible solution to support other CI tools
- Works on standard on premise (VMWare, Open Stack) or cloud infrastructure (AWS, Rack Space, etc)
- Pluggable with most CMP solutions for cost analytics and governance

Build-On: Technology Stack



Simplify creation of Scalable CI Containers



1

Configure CI Master

Leverage existing Jenkins Master and Jobs

2

Place Build Requests

Jobs or requests are triggered based on identified events

3

Dynamic Slave are spawned

Builds are executed on containers running as slave nodes

4

Pipeline Execution

Slaves executes pipeline stages

5

Jobs executed automatically

Post build console logs are archived and presented as dashboards

6

Teardown

After build completion, slave containers are destroyed

Build Pipeline as Code



1

Define Pipeline as Code

Descriptively define jobs in YAML files

2

Check-in Code

Focus on your actual software development

3

Auto run CI

Check in automatically detected and CI is triggered

4

Pipeline Execution

Headless CI executes pipeline stages

5

Dashboards

Post build console logs are archived and presented as dashboards

6

Teardown

After build completion, headless CI containers are destroyed

Benefits of Build-On

Scalable CI Set-up for Build Engineer



Unlimited On-Demand Scalability



Low Maintenance



Easy Configuration and updates

Pipeline-as-Code for Developer



Headless CI that scales on demand



Codified pipelines increases productivity



Improves feedback and quality by avoiding queues

Next Gen developer friendly CI solution that scales on demand, enabling focus on delivering quality code, faster.

Resources

Basics
Challenges of
Build
Engineer

Basics
Challenges of
Developer

Demo Video
Build as a
service

Demo Video
Pipeline as
code

Visit us online at www.cognizant.com or follow us on Twitter: @Cognizant

Cognizant (NASDAQ: CTSI) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 100 development and delivery centers worldwide and approximately 217,700 employees as of March 31, 2015, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world.

None of this document may be reproduced, stored in a retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the express written permission from Cognizant. The information contained herein is confidential and proprietary to Cognizant. All other trademarks mentioned herein are the property of their respective owners.



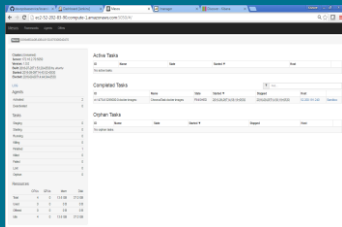
KEEP CHALLENGING™



Appendix – Screen Shots

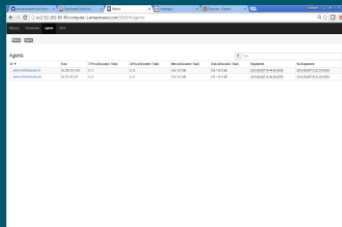
Deliver Continuous Integration in a Box

**Mesos UI –
No Jobs are running**



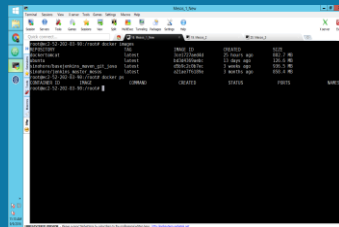
The screenshot shows the Mesos UI interface. The 'Active Tasks' table is empty. The 'Completed Tasks' table shows one task that has finished. The 'Cluster Tasks' table is also empty.

**Shows the list of
Mesos slave.**



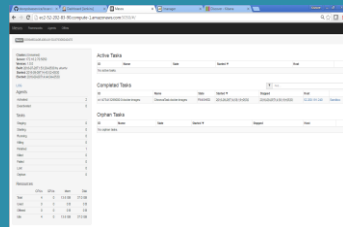
The screenshot shows the 'Slaves' table in the Mesos UI. It lists several Mesos slaves with their IDs, roles, and resource profiles.

**List of docker images and
running containers**



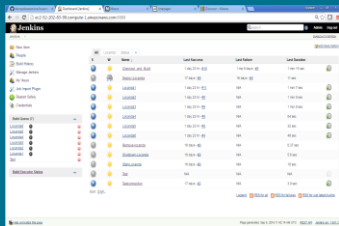
The screenshot shows the Docker UI interface. It displays a list of running containers with their names, IDs, and status.

**Dockerized Jenkins has
predefined plugins and
configurations**



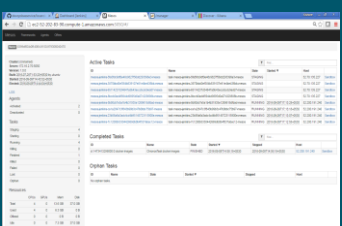
The screenshot shows the Jenkins UI interface. It displays the 'Active Tasks' and 'Completed Tasks' tables, which are empty.

**Jenkins started at 8888 port
Triggering Builds
note Zero executors**



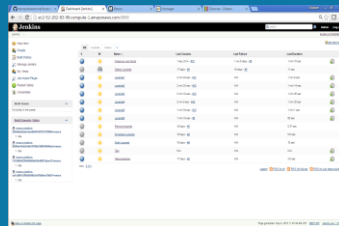
The screenshot shows the Jenkins UI interface. The 'Executors' section shows zero executors. The 'Builds' section shows a list of builds.

**On-demand slaves are spinning
up. Mesos run the slave Jenkins
as containers**



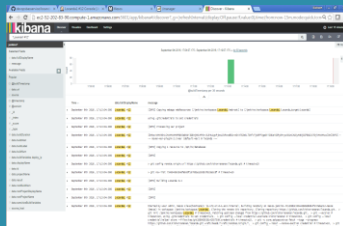
The screenshot shows the Mesos UI interface. It displays a list of Mesos slaves, including those that are spinning up.

**Jobs has been run and the
slaves are getting terminated.
Utilization returns to normal**



The screenshot shows the Jenkins UI interface. It displays a list of builds, including those that are running and those that are terminated.

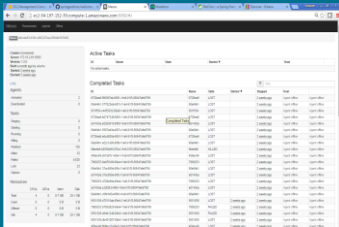
**Logs of the job executed in
Kibana**



The screenshot shows the Kibana UI interface. It displays a list of logs, including those from the Jenkins job.

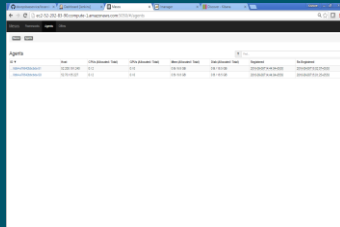
Generate Build Pipeline as Code

Mesos UI – Mesos Master showing all resources in the Idle state, ready for utilization



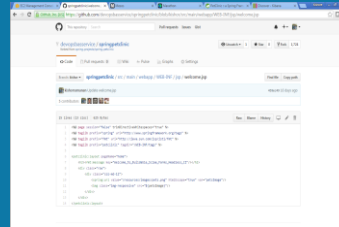
Resource ID	State	Owner	Labels	CPUs	Memory	Disk
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0

Shows the list of Mesos slave.



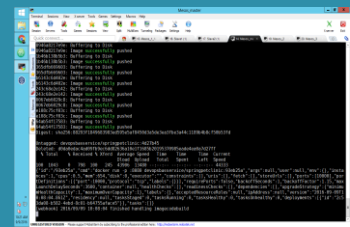
Slave ID	State	Owner	Labels	CPUs	Memory	Disk
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0
mesos-slave-00000000000000000000	Idle			1.0	1.0	1.0

Original code in GitHub Repo Check H2 tag, it is same as the app deployment title



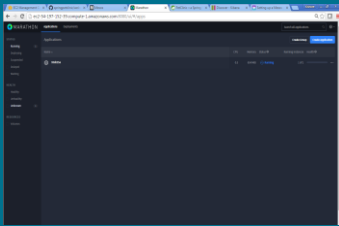
```
def deploy(app_name, version):
    # Deploy the application to the Mesos cluster
    # This function will be called by the Jenkins job
    # The app_name and version are passed as arguments
    # The function will return the status of the deployment
```

Git-Hub Triggers Web-hook and completes task



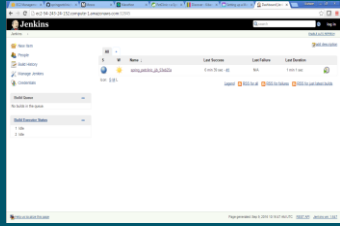
```
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
jenkins-jobs-00000000000000000000
```

Marathon showing where the headless CI is running



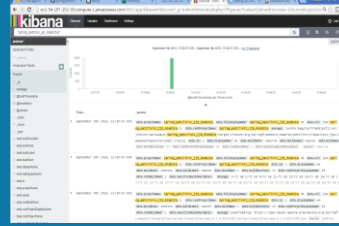
Application ID	State	Owner	Labels	CPUs	Memory	Disk
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Running			1.0	1.0	1.0

Jenkins started in the port

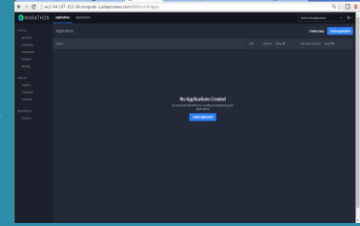


Job Name	Status	Build Number	Build Time	Build Duration
jenkins-jobs-00000000000000000000	Success	1	2016-01-01 10:00:00	10:00:00
jenkins-jobs-00000000000000000000	Success	2	2016-01-01 10:05:00	10:05:00
jenkins-jobs-00000000000000000000	Success	3	2016-01-01 10:10:00	10:10:00
jenkins-jobs-00000000000000000000	Success	4	2016-01-01 10:15:00	10:15:00
jenkins-jobs-00000000000000000000	Success	5	2016-01-01 10:20:00	10:20:00
jenkins-jobs-00000000000000000000	Success	6	2016-01-01 10:25:00	10:25:00
jenkins-jobs-00000000000000000000	Success	7	2016-01-01 10:30:00	10:30:00
jenkins-jobs-00000000000000000000	Success	8	2016-01-01 10:35:00	10:35:00
jenkins-jobs-00000000000000000000	Success	9	2016-01-01 10:40:00	10:40:00
jenkins-jobs-00000000000000000000	Success	10	2016-01-01 10:45:00	10:45:00

Logs in Kibana searched by the Job name



Jobs get terminated in marathon



Application ID	State	Owner	Labels	CPUs	Memory	Disk
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0
jenkins-jobs-00000000000000000000	Terminated			1.0	1.0	1.0