Jenkins: A complete solution

From Continuous Integration to Continuous Delivery

For

HSBC

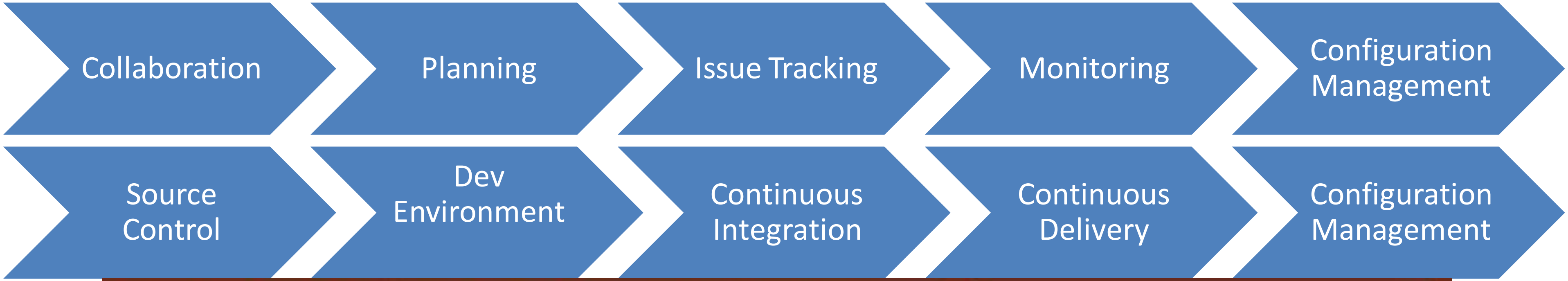
Rajesh Kumar

DevOps Architect

@RajeshKumarIN | www.RajeshKumar.xyz

Agenda

- □ Why Jenkins?
- □ Introduction and some facts about Jenkins
- □ Supported tech stacks and platforms
- □ Security
- □ Leveraging Jenkins across various projects
- □ Enabling continuous delivery
- □ Best practices



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

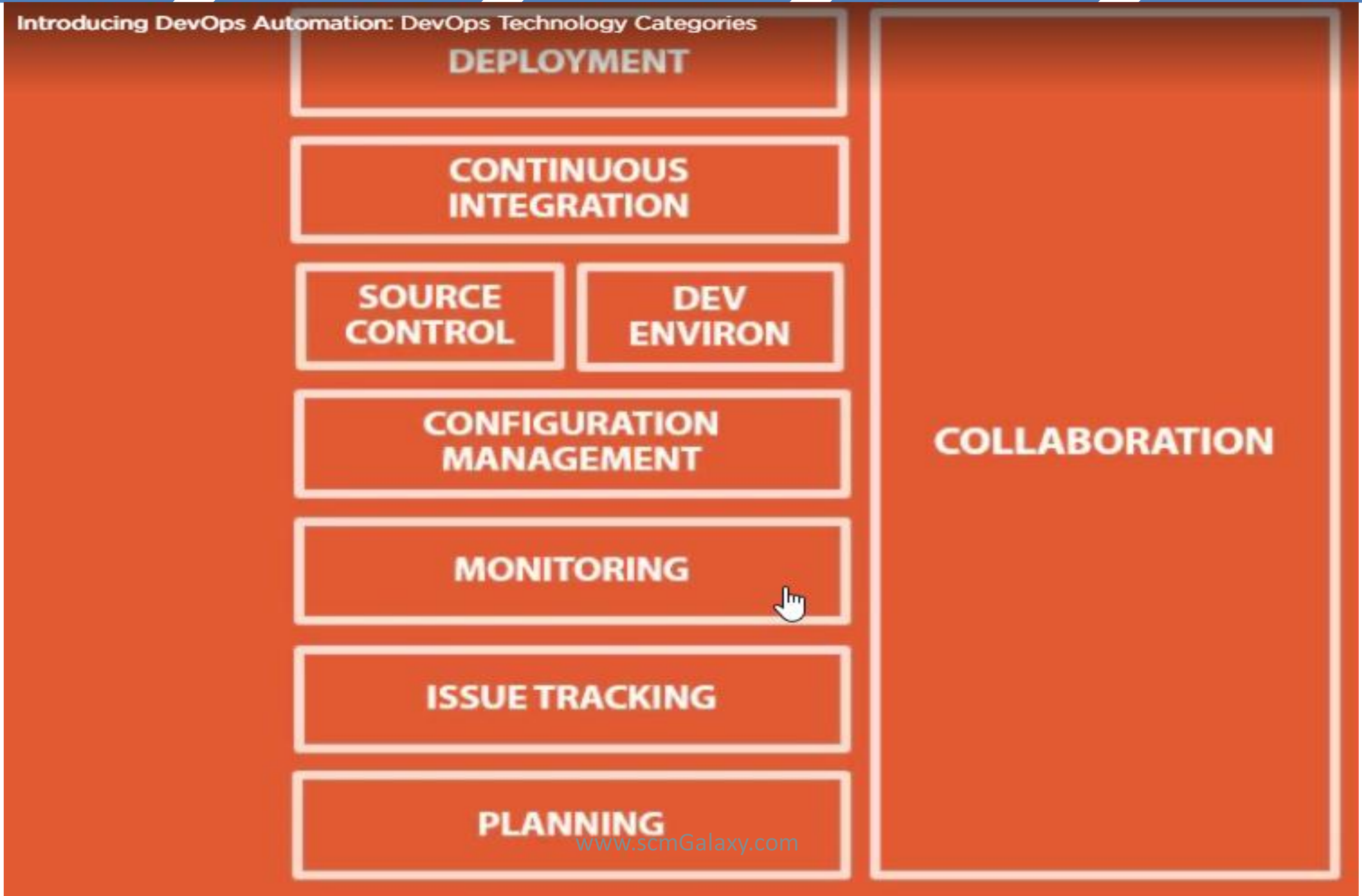
Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Configuration Management



Introducing DevOps Automation: DevOps Technology Categories

DEPLOYMENT

CONTINUOUS INTEGRATION

SOURCE CONTROL

DEV ENVIRON

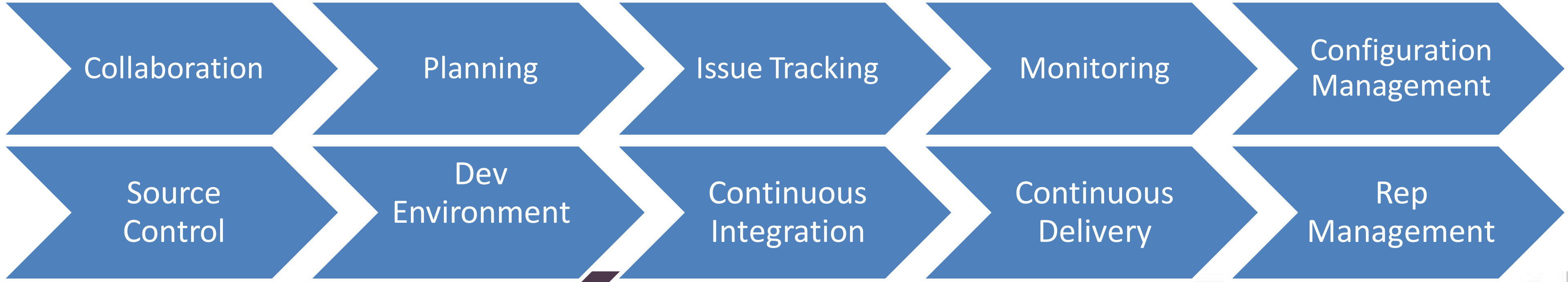
CONFIGURATION MANAGEMENT

MONITORING

ISSUE TRACKING

PLANNING

COLLABORATION



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

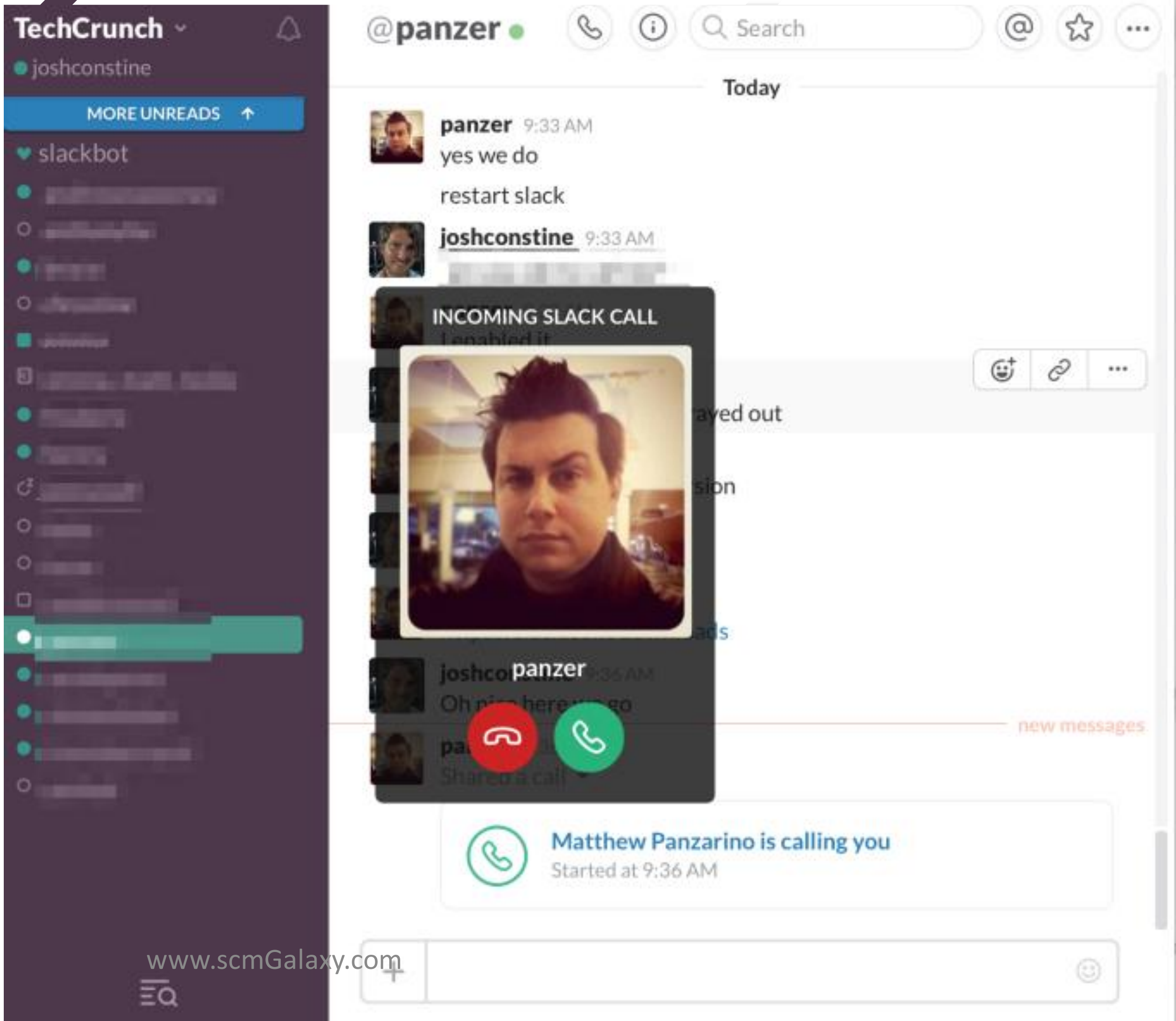
Dev Environment

Continuous Integration

Continuous Delivery

Rep Management

SLACK



Today

panzer 9:33 AM
yes we do

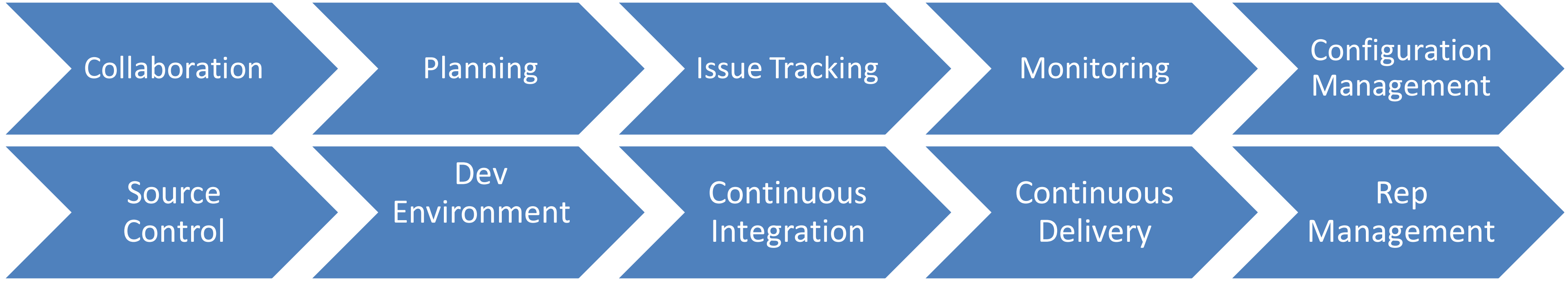
restart slack

joshconstine 9:33 AM

INCOMING SLACK CALL

panzer

Matthew Panzarino is calling you
Started at 9:36 AM



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

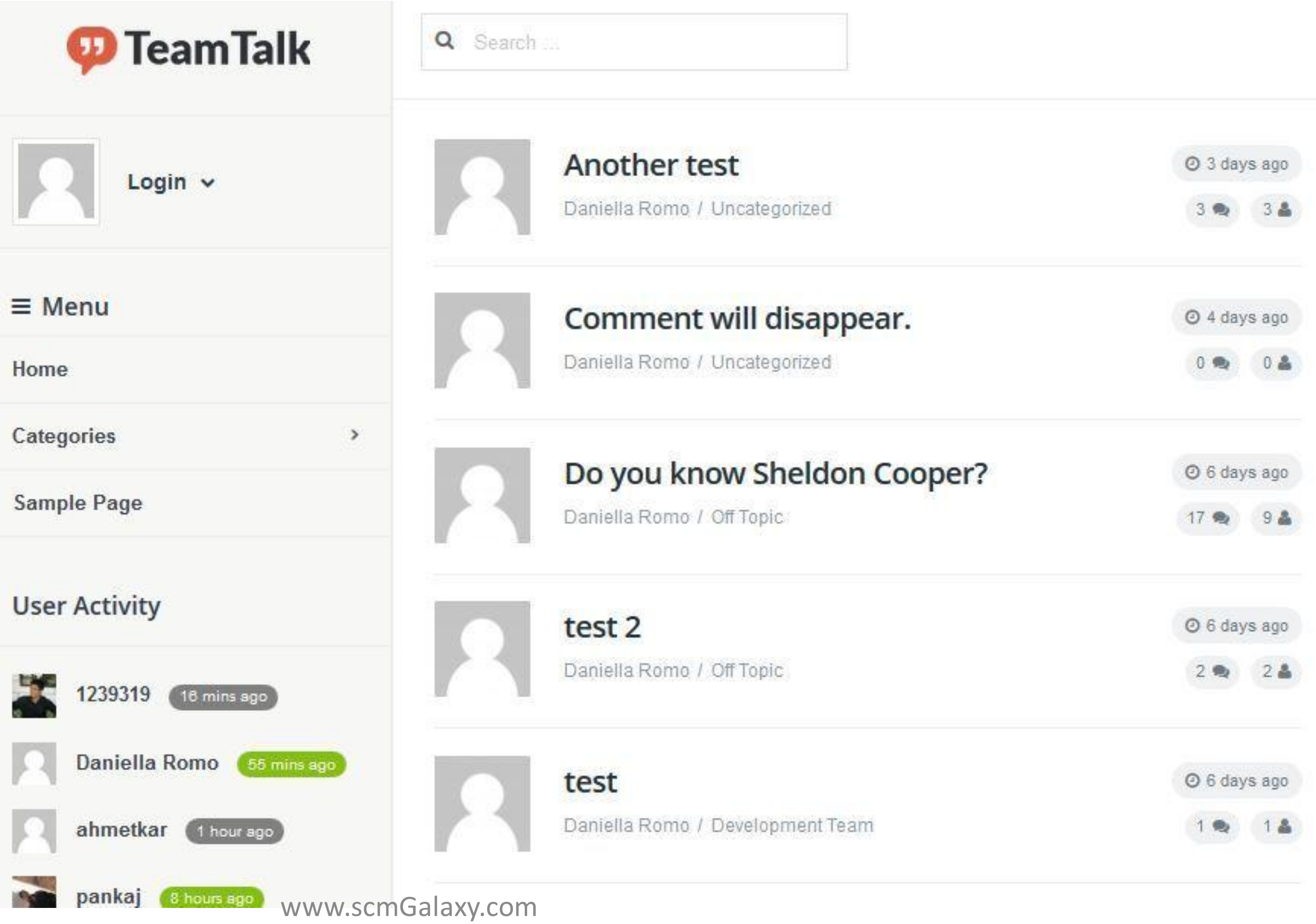
Dev Environment

Continuous Integration

Continuous Delivery

Rep Management

WORD PRESS



TeamTalk

Search ...

Login

≡ Menu

Home

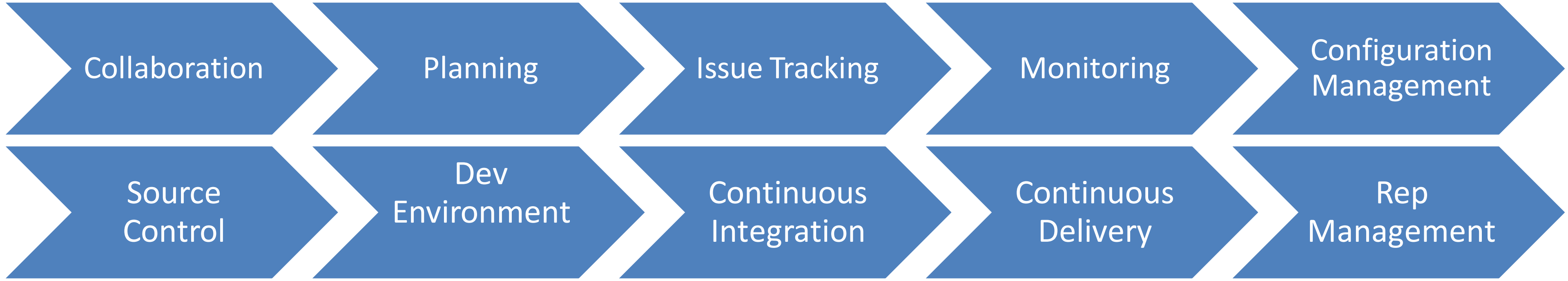
Categories ›

Sample Page

User Activity

1239319	16 mins ago
Daniella Romo	55 mins ago
ahmetkar	1 hour ago
pankaj	8 hours ago

Topic	Posted by / Category	Age	Replies	Participants
Another test	Daniella Romo / Uncategorized	3 days ago	3	3
Comment will disappear.	Daniella Romo / Uncategorized	4 days ago	0	0
Do you know Sheldon Cooper?	Daniella Romo / Off Topic	6 days ago	17	9
test 2	Daniella Romo / Off Topic	6 days ago	2	2
test	Daniella Romo / Development Team	6 days ago	1	1



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

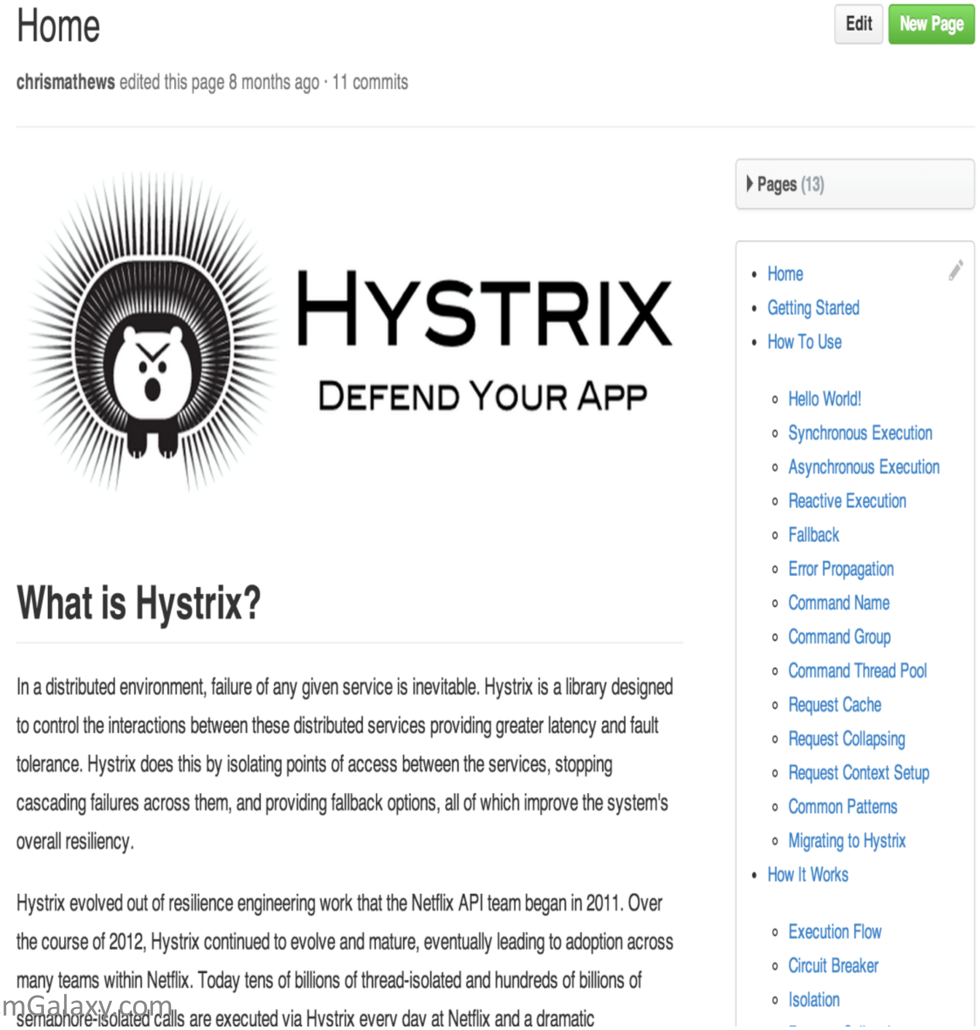
Dev Environment

Continuous Integration

Continuous Delivery

Rep Management

GITHUB WIKI



Home

Edit | New Page

chrismathews edited this page 8 months ago · 11 commits

HYSTRIX

DEFEND YOUR APP

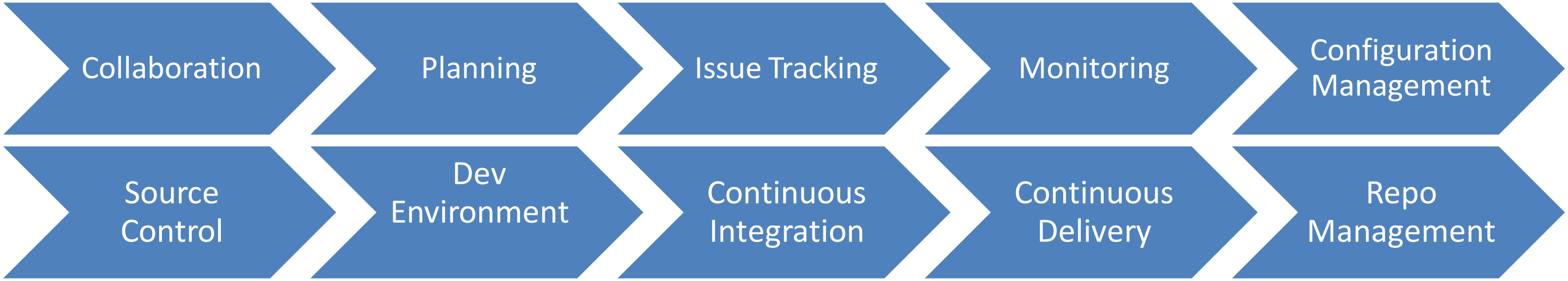
What is Hystrix?

In a distributed environment, failure of any given service is inevitable. Hystrix is a library designed to control the interactions between these distributed services providing greater latency and fault tolerance. Hystrix does this by isolating points of access between the services, stopping cascading failures across them, and providing fallback options, all of which improve the system's overall resiliency.

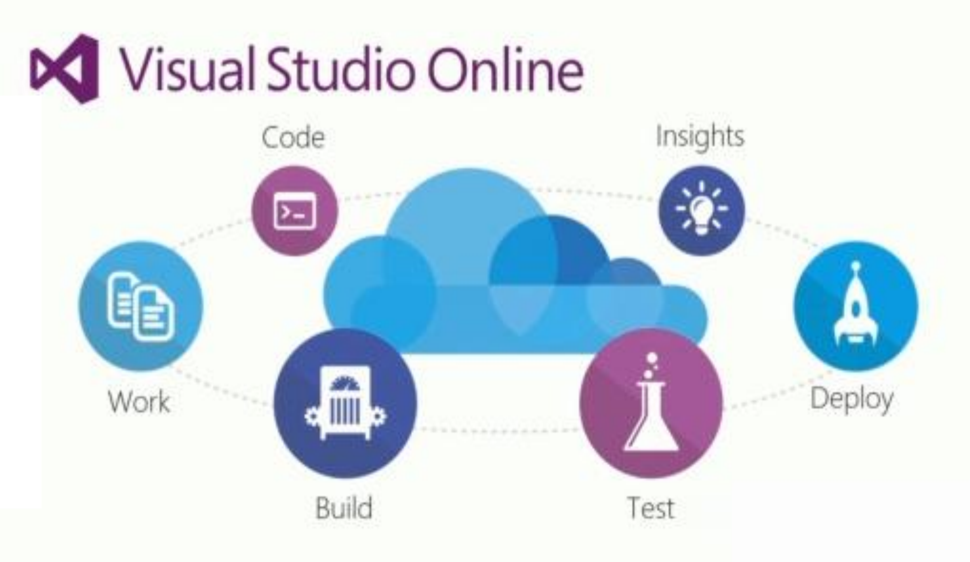
Hystrix evolved out of resilience engineering work that the Netflix API team began in 2011. Over the course of 2012, Hystrix continued to evolve and mature, eventually leading to adoption across many teams within Netflix. Today tens of billions of thread-isolated and hundreds of billions of semaphore-isolated calls are executed via Hystrix every day at Netflix and a dramatic

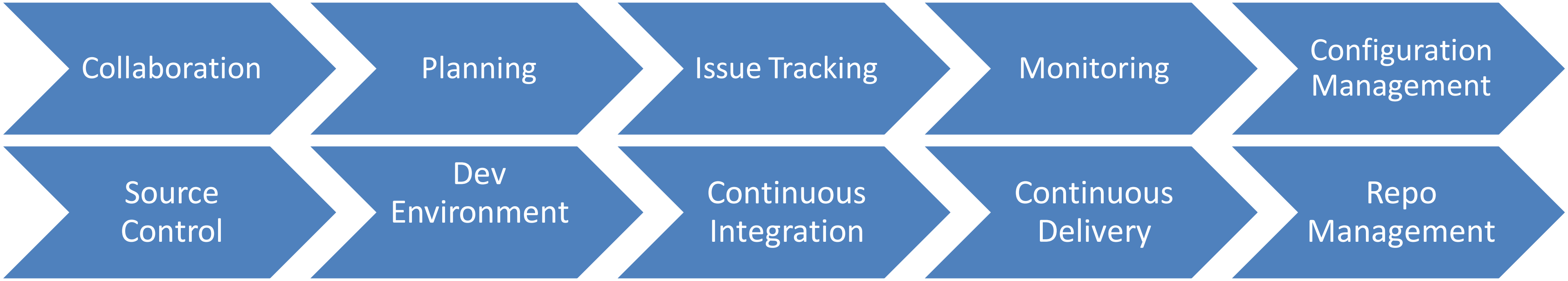
▸ Pages (13)

- Home
- Getting Started
- How To Use
    - Hello World!
    - Synchronous Execution
    - Asynchronous Execution
    - Reactive Execution
    - Fallback
    - Error Propagation
    - Command Name
    - Command Group
    - Command Thread Pool
    - Request Cache
    - Request Collapsing
    - Request Context Setup
    - Common Patterns
    - Migrating to Hystrix
- How It Works
    - Execution Flow
    - Circuit Breaker
    - Isolation


Collaboration
Planning
Issue Tracking
Monitoring
Configuration Management
Source Control
Dev Environment
Continuous Integration
Continuous Delivery
Repo Management

Your entire project, in a single glance.





Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

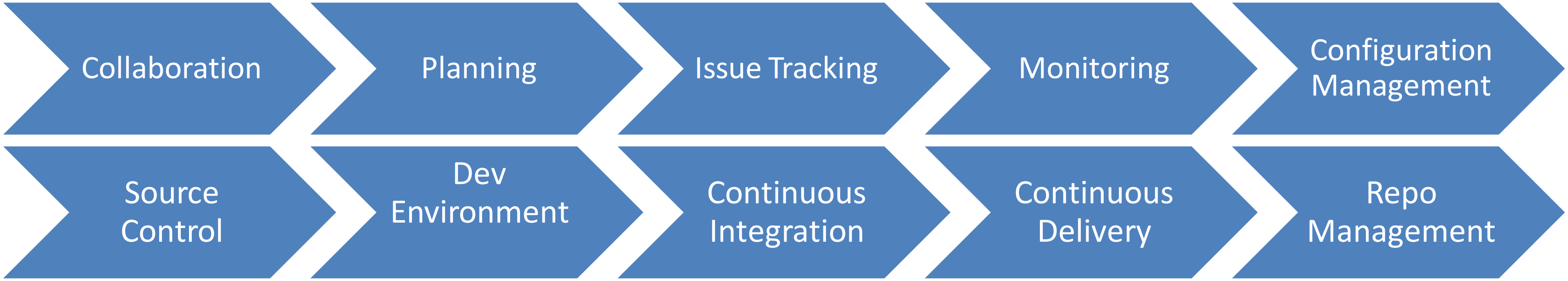
Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Repo Management



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Repo Management

logstash

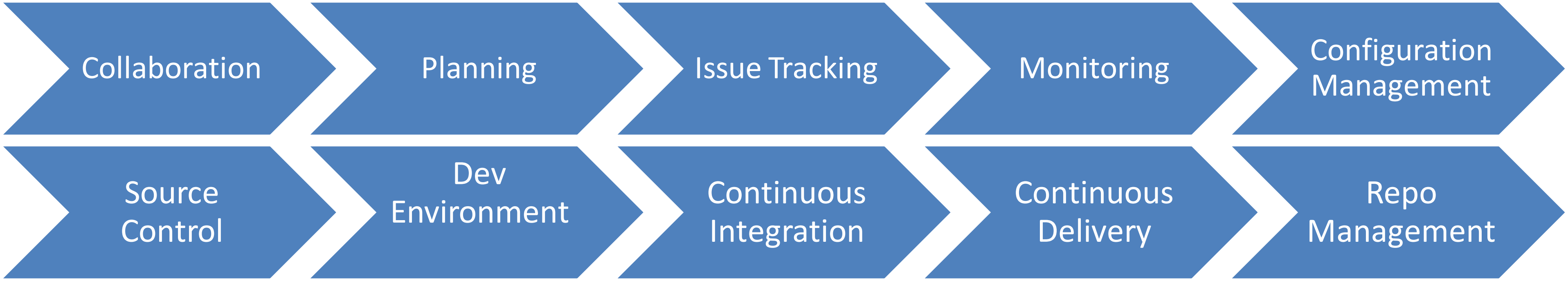
Microsoft® System Center™

kibana

STATSD

graphite

New Relic®



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

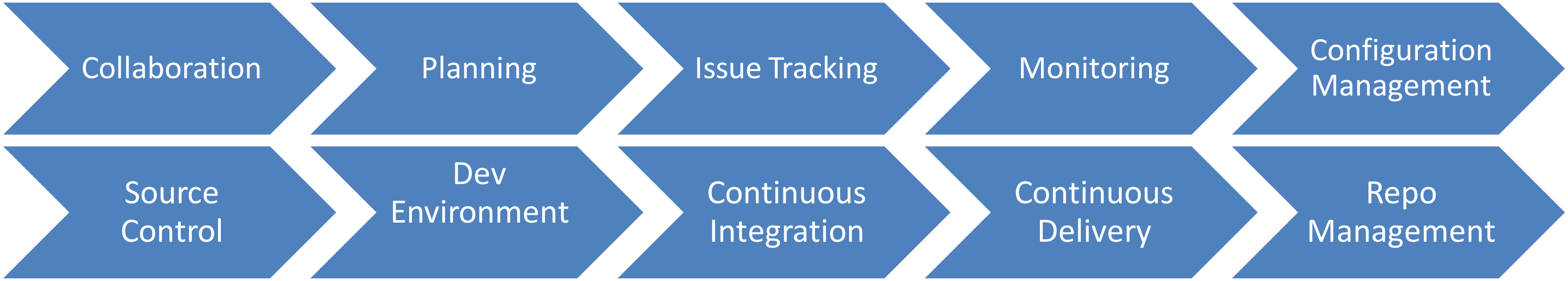
Dev Environment

Continuous Integration

Continuous Delivery

Repo Management

CFEngine
puppet labs®
PalletOps
ANSIBLE
CHEF™
SALTSTACK


Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

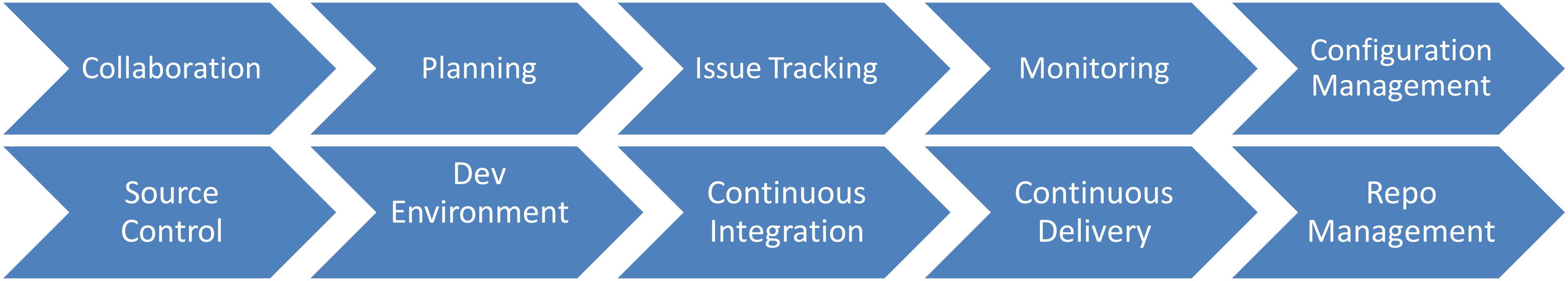
Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Repo Management



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

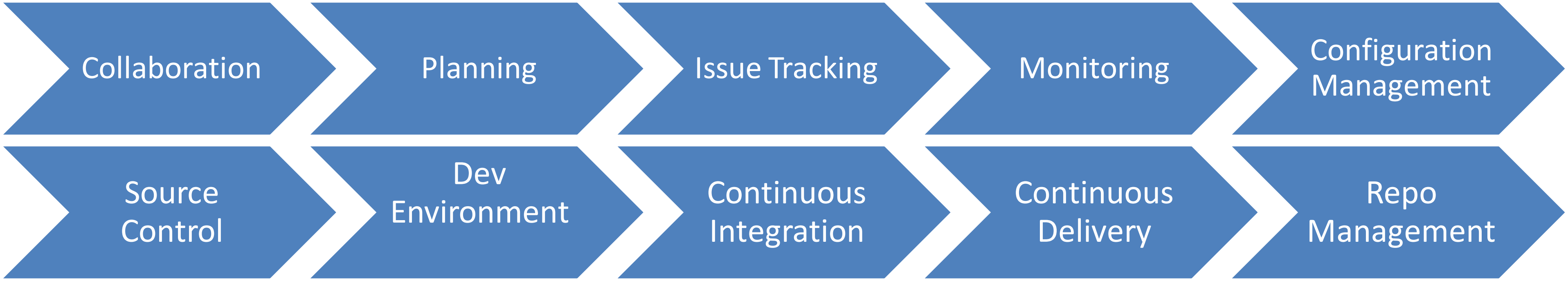
Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Repo Management



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

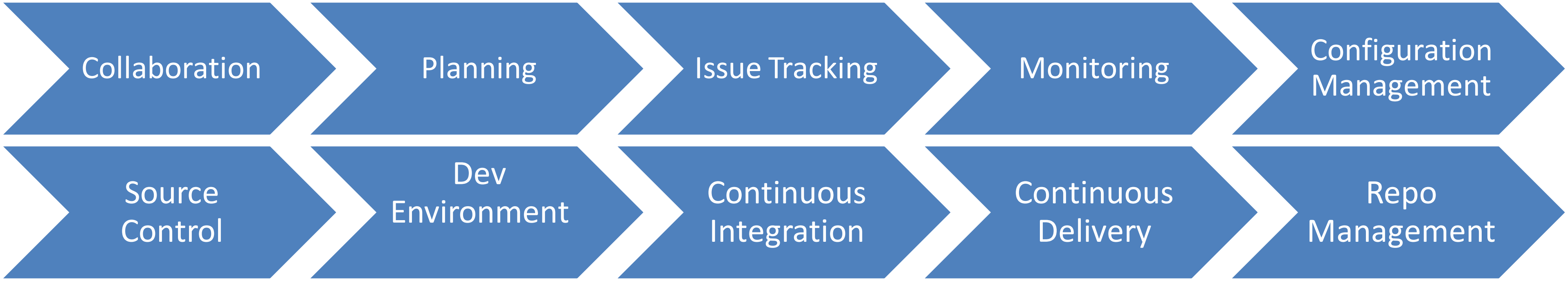
Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Repo Management

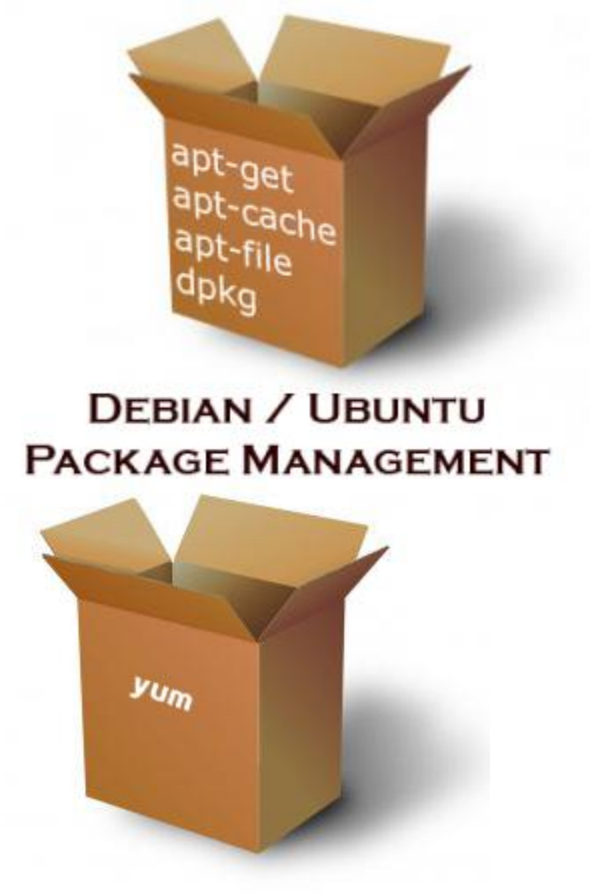


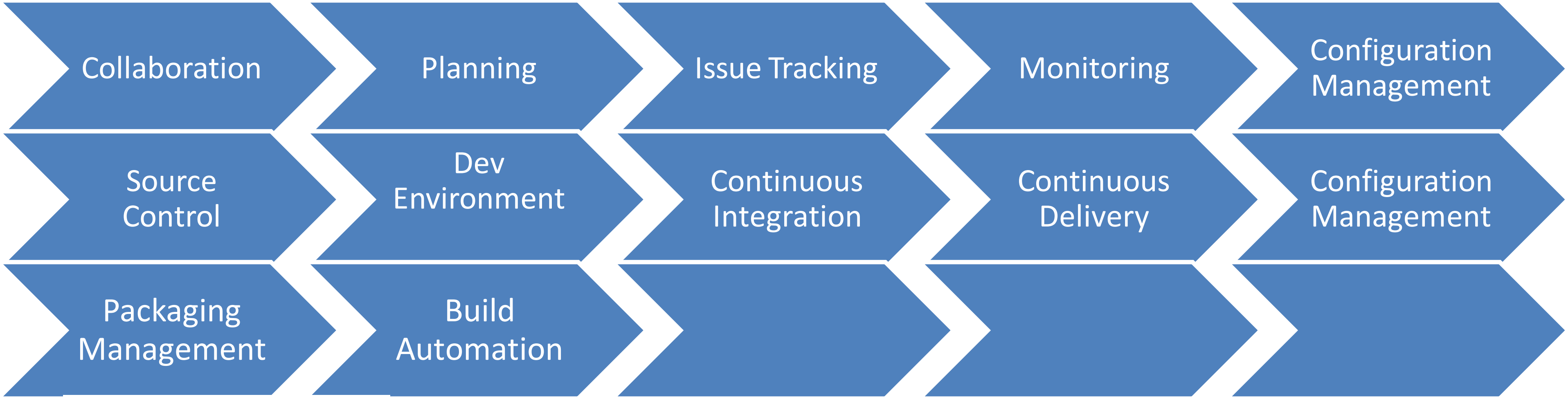
Nexus

artifactory

archiva™

nuget

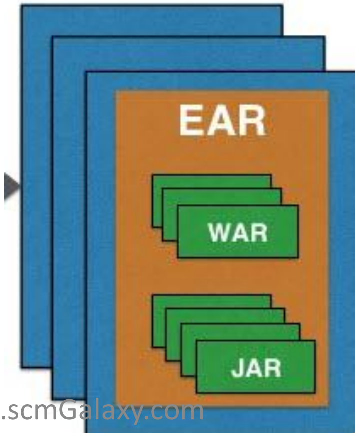



Collaboration
Planning
Issue Tracking
Monitoring
Configuration Management
Source Control
Dev Environment
Continuous Integration
Continuous Delivery
Configuration Management
Packaging Management
Build Automation

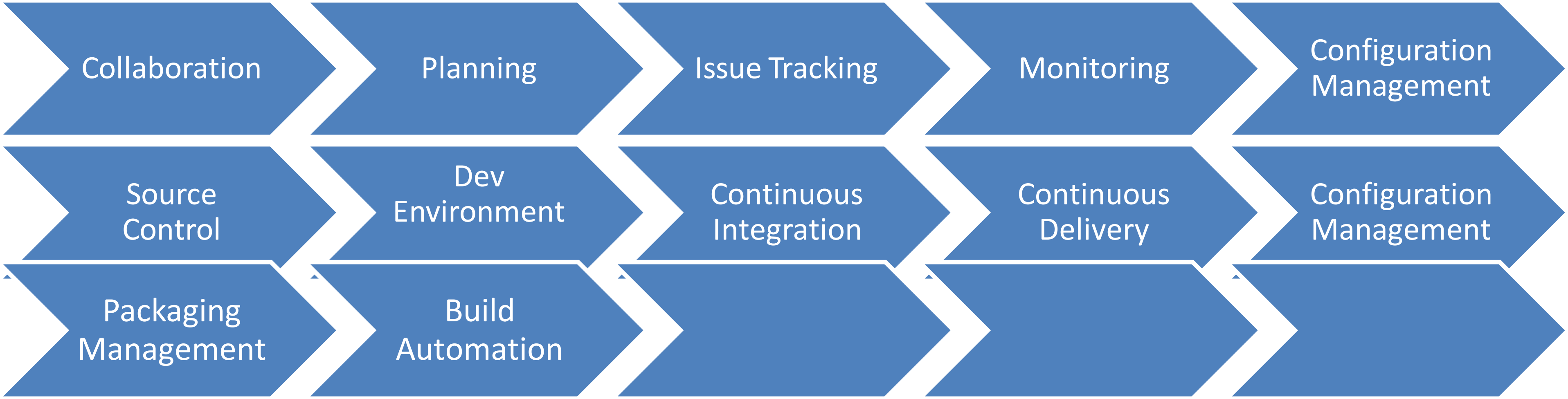
InstallShield

DEBIAN / UBUNTU
ACKAGE MANAGEMENT

FLEXERA SOFTWARE®
InstallAnywhere®



nuget



Collaboration

Planning

Issue Tracking

Monitoring

Configuration Management

Source Control

Dev Environment

Continuous Integration

Continuous Delivery

Configuration Management

Packaging Management

Build Automation

Managing Projects with
GNU Make
maven
PACHE ANT>
MSBuild
Nmake
Make

HOW TO INTEGRATE EVERYTHING?

Jenkins

TeamCity

BAMBOO

Travis CI

Jenkins: Introduction and facts

Jenkins is an award-winning application that monitors executions of repeated jobs, such as building a software project or jobs run by cron. Among those things, current Jenkins focuses on the following two jobs:

- Building/testing software projects continuously
- Monitoring executions of externally-run jobs

Facts:

- Written in Java and initially was supposed to be used as a CI tool
- Over 600 plugins to customize Jenkins as per your need
- Over 1000+ public repositories on Github, 500+ contributors, strong commit activity
- Free open source and most widely used tool for maintaining continuous integration cycle. Google trend says it all

Supported tech stacks and platforms

Other popular non java projects supported by Jenkins:

- .Net
- Ruby
- PHP
- Drupal
- Perl
- C++
- Node.js
- Python
- Android
- Scala

Platforms supported by Jenkins:

- □ Windows
- □ Ubuntu/Debian
- □ Red Hat/Fedora/CentOS
- □ Mac OS X
- □ openSUSE
- □ FreeBSD
- □ OpenBSD
- □ Solaris/OpenIndiana
- □ Gentoo

Feature

❑ Easy install, easy upgrade, easy configuration

❑ Distributed builds – Arguably most powerful feature.

❑ Monitoring external jobs

❑ No limit to the number of jobs, number of slave nodes

❑ Plugin architecture: Support for various version control systems, authentication methods, notification, workflow building, and many more features can be added.

❑ Jenkins provides machine-consumable remote access API to its functionalities

❑ Actually there are lot of useful plugins. The list is too long to mention here. Go on, explore on your own. There’s plugin available for almost everything you would want.

Securing Jenkins

In the default configuration, Jenkins does not perform any security check. This means any person accessing the website can configure Jenkins and jobs, and perform builds. While this configuration is acceptable during initial evaluation of the software, Jenkins should be configured to authenticate users and enforce access control in most other situations, especially when exposed to the Internet.

This article explains in detail how to secure your Jenkins.

- ❑ What I usually do is: As an administrator set up ‘Project-based Matrix Authorization Strategy’ and give only read right to users globally. At job level, you can give required rights to the users. This would help us create separate jobs for separate project teams on the same Jenkins instance.

Sharing Jenkins across projects

So here’s the use case: You are an Ops guy, maintaining the Jenkins Infrastructure and there are a lot of product teams wanting to use Jenkins for their continuous integration and delivery. Would you install a separate Jenkins instance for each team? Obviously No.

Also, each team should get access (after logging into Jenkins server) to view/run/modify only their project’s configured jobs. They shouldn’t be able to view anything else.

Everything mentioned above can be easily achieved with Jenkins:

- Depending upon the disk and resources usage of each project, you can decide whether the same master Jenkins can be used as a build server or you need a slave instance. This is the most powerful feature of Jenkins – Distributed builds.
- For restricting access to project teams, use ‘Project-based Matrix Authorization Strategy’ and create separate ‘views’ for each project. As described in the previous slide.

Enabling Continuous Delivery

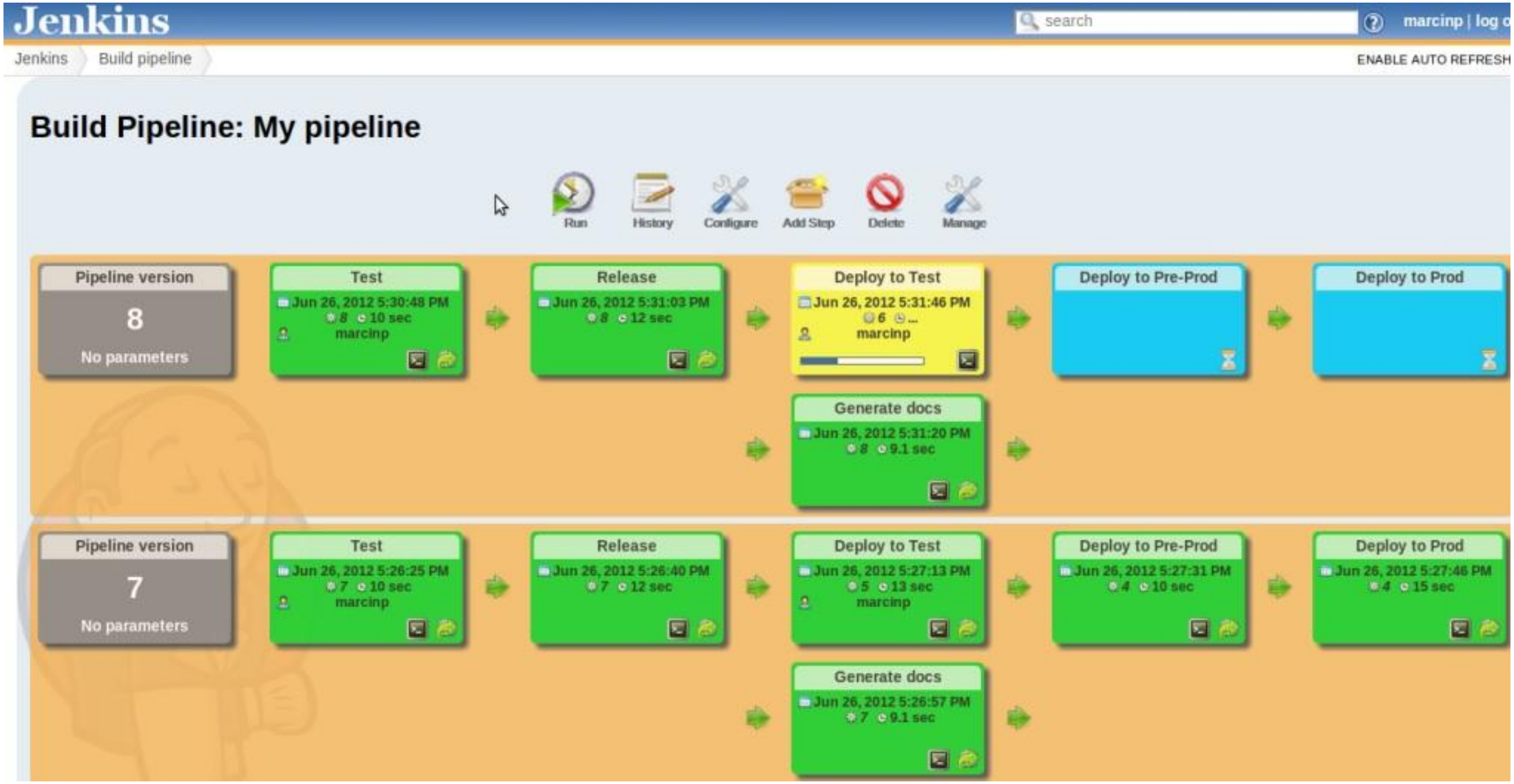
- Continuous Integration: It is the practice of merging development work with a Master/Trunk/Mainline branch constantly so that you can test changes, and test that changes work with other changes. The idea here is to test your code as often as possible to catch issues early. Most of the work is done by automated tests, and this technique requires a unit test framework. Typically there is a build server performing these tests, so developers can continue working while tests are being performed.

- Continuous Delivery: It is the continual delivery of code to an environment once the developer feels the code is ready to ship. This could be UAT or Staging or could be Production. But the idea is you are delivering code to a user base, whether it be QA or customers for continual review and inspection.

In other words, Continuous Delivery is a process that merges Continuous Integration with automated deployment, test, and release; creating a Continuous Delivery solution. Continuous Delivery doesn't mean every change is deployed to production ASAP. It means every change is proven to be deployable at any time. Check this article to get more insight.

Here, we would talk about enabling Continuous Delivery using Jenkins and it’s plugins. By using Build pipeline plugin in Jenkins, we can orchestrate the promotion of a version of software through quality gates and into production. By extending the concepts of CI you can create a chain of jobs each one subjecting your build to quality assurance steps. These QA steps may be a combination of manual and automated steps. Once a build has passed all these, it can be automatically deployed into production.

Sample build pipeline



Build Pipeline: My pipeline

Pipeline version
8
No parameters

Test
Jun 26, 2012 5:30:48 PM
8 10 sec
marcinp

Release
Jun 26, 2012 5:31:03 PM
8 12 sec

Deploy to Test
Jun 26, 2012 5:31:46 PM
6 ...
marcinp

Deploy to Pre-Prod

Deploy to Prod

Generate docs
Jun 26, 2012 5:31:20 PM
8 9.1 sec

Pipeline version
7
No parameters

Test
Jun 26, 2012 5:26:25 PM
7 10 sec
marcinp

Release
Jun 26, 2012 5:26:40 PM
7 12 sec

Deploy to Test
Jun 26, 2012 5:27:13 PM
5 13 sec
marcinp

Deploy to Pre-Prod
Jun 26, 2012 5:27:31 PM
4 10 sec

Deploy to Prod
Jun 26, 2012 5:27:46 PM
4 15 sec

Generate docs
Jun 26, 2012 5:26:57 PM
7 9.1 sec

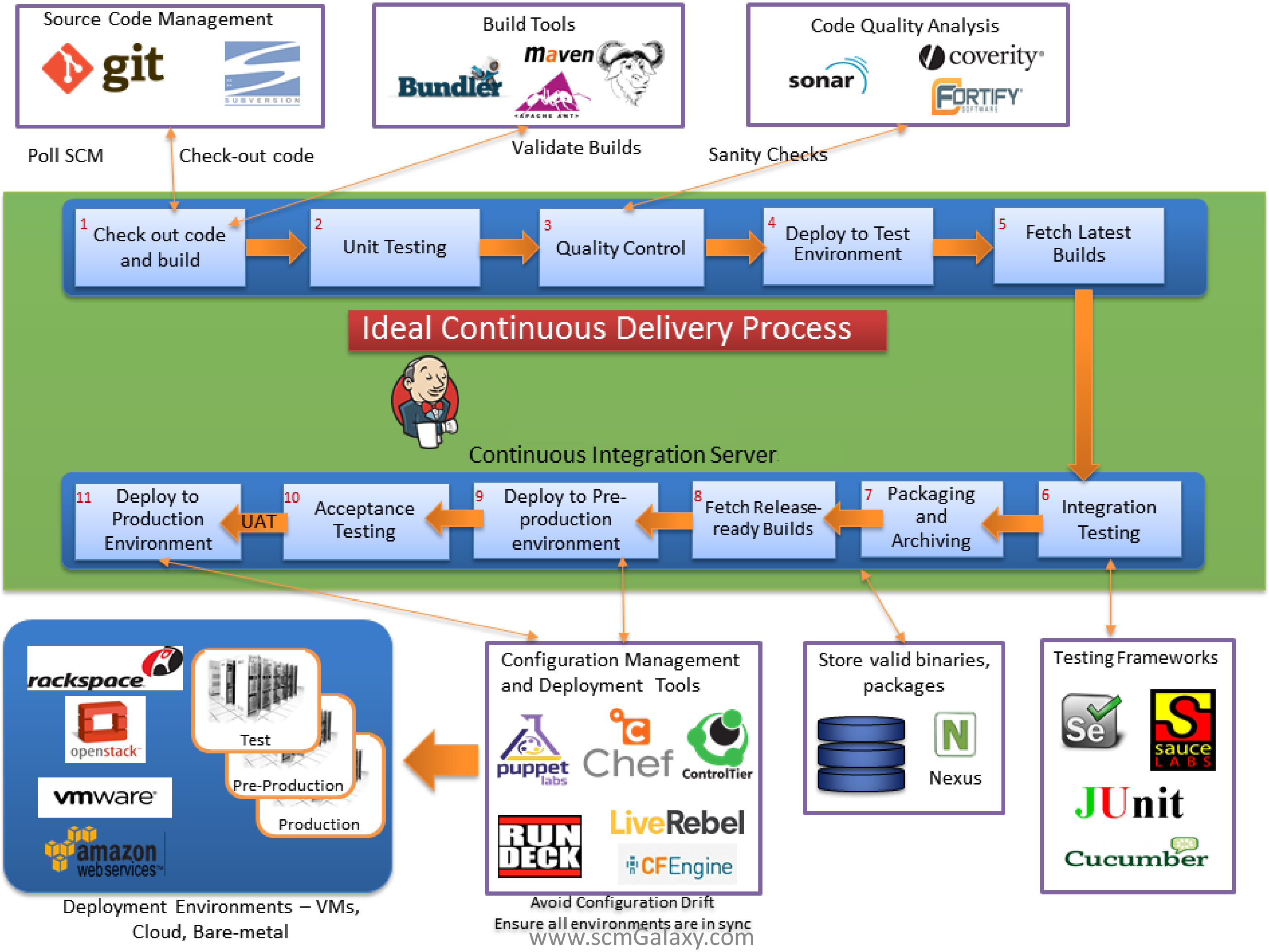
Phases of Continuous Delivery

- □ Unit Test
- □ Code Quality Analysis
- □ Deploy to Test Environment
- □ Integration Test
- □ Packaging and Archiving
- □ Deploy to Preproduction Environment
- □ Acceptance Test
- □ Deploy to Production Environment

Jenkins has every plugin required for the ideal Continuous Delivery process, that too free of cost.

With the help of Jenkins, we can create customized build pipeline to create a dashboard of our own and enable Continuous Delivery in easy steps

Continuous Delivery Process



Source Code Management

Build Tools

Code Quality Analysis

Poll SCM

Check-out code

Validate Builds

Sanity Checks

1 Check out code and build

2 Unit Testing

3 Quality Control

4 Deploy to Test Environment

5 Fetch Latest Builds

Ideal Continuous Delivery Process

Continuous Integration Server

6 Integration Testing

7 Packaging and Archiving

8 Fetch Release-ready Builds

9 Deploy to Pre-production environment

10 Acceptance Testing

UAT

11 Deploy to Production Environment

Test

Pre-Production

Production

Deployment Environments – VMs, Cloud, Bare-metal

Configuration Management and Deployment Tools

Avoid Configuration Drift

Ensure all environments are in sync

Store valid binaries, packages

Nexus

Testing Frameworks

Best Practices

- □ Make sure you have backups – better late than never
- □ Plan disk usage – make sure it’s expandable
- □ For easier installation and migration, use native packages if possible
- □ Do distributed builds
- □ Use labels to optimize resource utilization and improve manageability
- □ Make your Jenkins URL short and memorable
- □ Discard old build records to keep your Jenkins instance healthy

Check this Jenkins official page for best practices or download the white paper from the Jenkins founder Kohsuke Kawaguchi.

Questions?