DEVOPS IN PRACTICE

GOALS, PEOPLE AND TOOLS

ALESSIO CARMAZZI
SENIOR SOLUTION ARCHITECT

#RedHatOSD

WHO AM I?

- 17 years experience
- Sircle leader
- Senior System Engineer
- Senior Solution Architect
- Senior Pre-Sales Engineer
- DevOps Passionate

AGENDA

- ➔ Who Are you?
- ➔ In Practice (Genesis)
- ➔ Literature...eight steps
  - ➔ Businesss Goal
  - ➔ People
  - ➔ Tools
- ➔ The super customers

WHO ARE YOU?

- Your Company:...
- CULTURE:...
- Business Goals:... (WHY)
- People:... talents, skills, methodologies, processes (HOW)
- Tools: ... (WHAT)

PURPOSE of the presentation

IN PRACTICE (SORINT DevOps PERSPECTIVE)

- GOALS
  - Business Outcome: Agility, Cost saving, Flexibility, scalability..
- PEOPLE
  - CULTURE, Methodology, processes, skills...
- TOOLS
  - Automation, CI/CD, TestFirst, TestEverything, ...

IN PRACTICE

The truth, not dreams or lies!

The Genesis of this presentation is the SUM of all our experiences with REAL customers

BANK

TELCO

FINTECH

SORINT lab

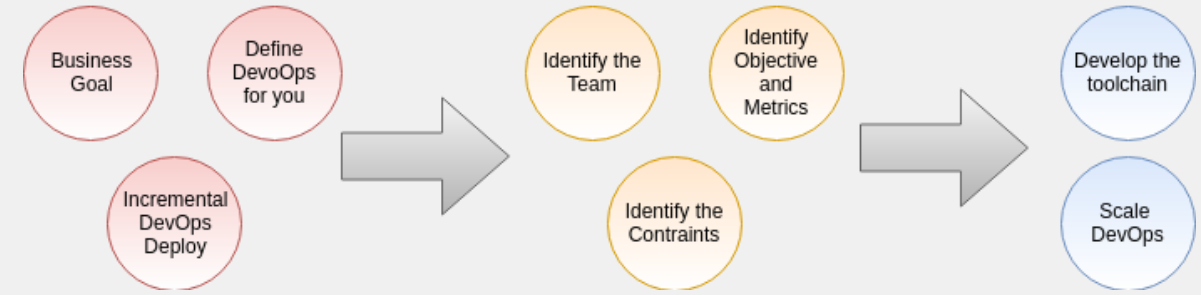
UTILITIES

STARTUP

… literature…

- **PASSIONATE** Best practices in literature
  - Readed Book, Blog, Articles
- **PASSIONATE** Investigation on Success Story with REAL customer
  - Collapsed our experiences in a Sorint.LAB Bok (Book of Knowledge)

... literature... eight steps



Business Goals

- Using Agile Methodology in order to get RAPID FEEDBACK
- VALUE throught:
  - “Welcomming Changes”
  - “Adopt and Adapt”
  - What about “Maintainability”?
- Go to market throught “Rapid Delivery” and “continuous Delivery”
- Rapid Changes… Microservices… Containers…

We need DEVOPS!

People

People are the main ingredient in a successful DevOps initiative
Teaching technical skills is easier than teaching the
correct behaviors, and the wrong behaviors will derail the DevOps effort.

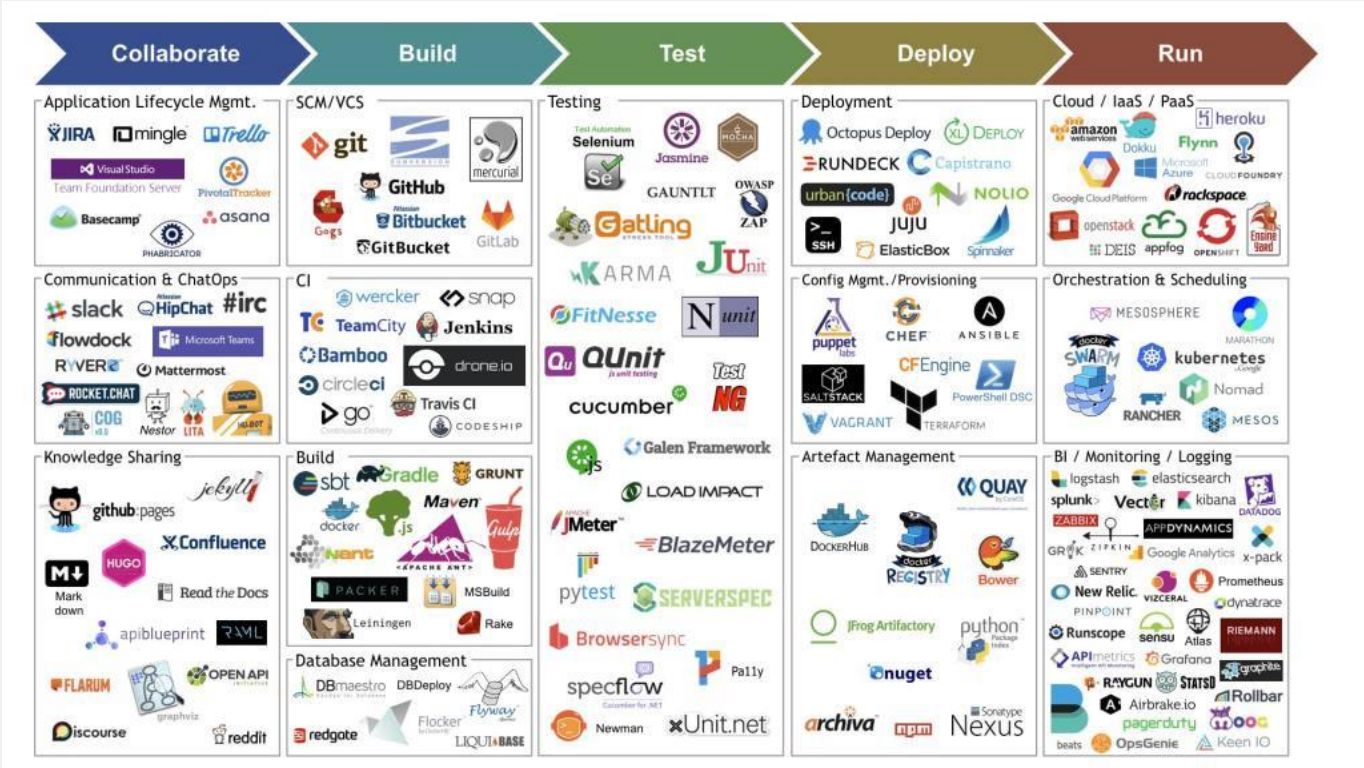
Behaviors to look for include the following:

- Team player
- Motivated
- Smart
- Understands risks
- Lifelong learner

Behavior patterns to avoid include the following:

- Poor people skills
- Followers
- Irrational risk takers
- Mediocrity

Tools...landscape



The Super CUSTOMER

GOALS

- Write software using standard
- Manage Activities using Agile Methodologies
- Adopt automation for CI/CD
- Application Modernization, Refactoring to Microserviecs
- Quality Assurance to write Quality code

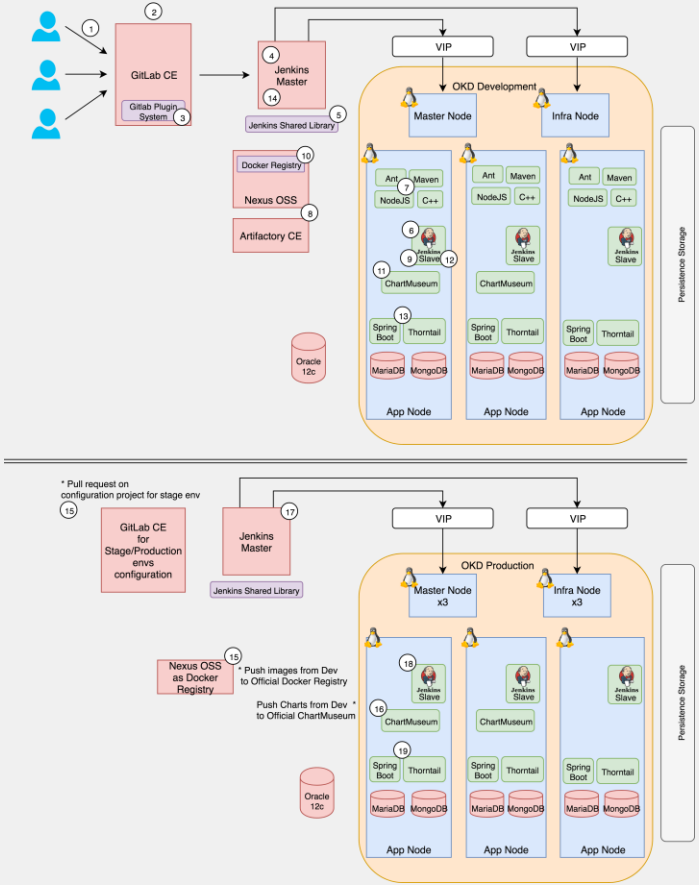
PEOPLE

- Adapt the Dev concept to the Ops environment
- Skill: Refocus to define application build mode to make it repeatable
- 12(+3) Factor Adoption
- Smart Framework introduction (Spring Boot, Wildfly Swarm, GO)
- Progressive adjustment to the Quality Assurance of the code

TOOLS

- Gitlab, Jenkins (master/slave, shared libraries) to compile on different platform, ant, maven, gradle, npm,..artifactory, chart museum, monocular, nexus, openshift, docker, helm ...

TOOLS ARCHITECTURE


GitLab CE
Gitlab Plugin System
Jenkins Master
Jenkins Shared Library
Docker Registry
Nexus OSS
Artifactory CE
VIP
VIP
OKD Development
Master Node
Infra Node
Ant
Maven
NodeJS
C++
Jenkins Slave
ChartMuseum
Spring Boot
Thorntail
MariaDB
MongoDB
App Node
Persistence Storage
Oracle 12c
* Pull request on configuration project for stage env
GitLab CE for Stage/Production envs configuration
Jenkins Master
Jenkins Shared Library
VIP
VIP
OKD Production
Master Node x3
Infra Node x3
Nexus OSS as Docker Registry
* Push images from Dev to Official Docker Registry
Push Charts from Dev * to Official ChartMuseum
Oracle 12c
App Node
Persistence Storage

STEPS (Chart1)

1. Developers push their commits into Gitlab server
2. Gitlab verify message commit format and, if valid, accept the commit and notify Jenkins Master with payload received
3. Using Gitlab custom plugin, Gitlab check if job available on Jenkins following pushed project.
   If no job tracking exists, Gitlab invoke rest api to register a new job, based on custom job template
   If project already exists, Gitlab notify Jenkins payload.
4. Jenkins execute the build for received job
5. Job execution based on minimal Jenkinsfile (ex. maven job, nodejs job,etc) inherited from a customized shared library registered on Jenkins
   Jenkins shared library contains all steps required for a build, based on build type.
6. Jenkins communicate with Openshift develop cluster to spawn a Jenkins Slave. Jenkins Slave notify to Openshift a Maven, NodeJs or other deployment execution, based on custom docker images.
7. A pod started, checkout the project source code and execute build, unit test, QA test,etc
8. After project build, the shared pipeline execute artifact versioning on Artifactory or Nexus, depending on project type.
9. Jenkins slave invoke command to create application Docker image with tag related to short commit id
10. Jenkins slave push image to Nexus repository (used as docker image registry)
11. Jenkins slave produce a Chart based on Helm template and push produced Helm to ChartMuseum
12. Jenkins tells to Openshift cluster to install the new Helm and, after deployment, execute E2E test
13. Application deployed on a dedicated Openshift project (namespace)
14. Jenkins Master notified on build status

STEPS (Chart2)

15. If job executed on a master branch project, Jenkins Master pipeline create a pull request to Gitlab CE for Collaudo/Production. The pull request contains environments configuration for project.
Jenkins Master pipeline push Docker image to Collaudo/Production Docker Registry (Nexus)
16. Jenkins Master push HelmChart on Collaudo/Production Chartmuseum
17. On a dedicated Jenkins (Collaudo/Production) an authorised user merge the pull request and manually trigger Job Execution.
18. The Jenkins Job communicate to Openshift Collaudo/Production cluster to spawn a Jenkins Slave. Jenkins Slave execute the pipeline telling Openshift to install the new Helm and run E2E test
19. The new instance application is deployed in Openshift Collaudo/Production. If E2E tests failed, a rollback executed to restore previous version

SORINT RESOURCES

www.sorint.it
floatingpoint.sorint.it
www.downloadinnovation.it


SORINT lab
redhat.
GRAZIE PER L'ATTENZIONE
ALESSIO CARMAZZI
SENIOR SOLUTION ARCHITECT
#RedHatOSD