Riverstone
Technology Unlimited

DevOps Strategy Document

Contents

Contents of Figures

1. Objective

The objective of the document is to describe Riverstone's strategy for implementing and/or improving the DevOps practices and tools used across the various Development and Operations related processes at an organization. The areas that are suitable for automation are identified and possible solutions for each area are described. This is subject to change based on discussion with company management.

2. DevOps Overview

2.1. What is DevOps

DevOps can be defined as a culture or process or practise within an organization that increases the communication, collaboration and integration of the Development (which includes the QA team) and the Operations (IT Operations) teams. The aim is to automate and speed up the software delivery process much more frequently and reliably.

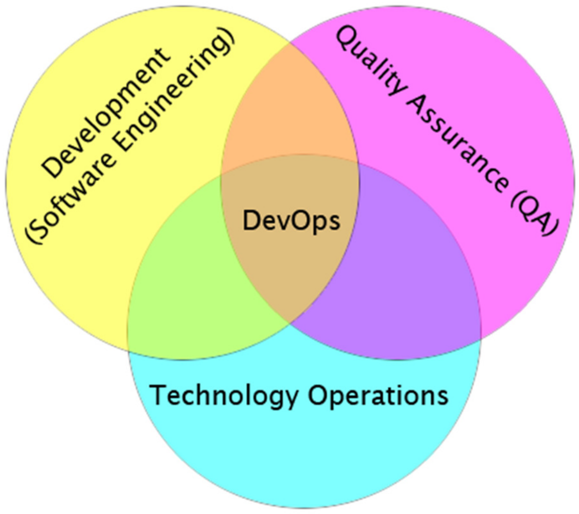


Figure 1 A Venn diagram way to understand DevOps

Even though 'Continuous Delivery' is another term that is used in conjunction with DevOps, there are some differences. The following diagram (Pic courtesy: collab.net) shows the relations

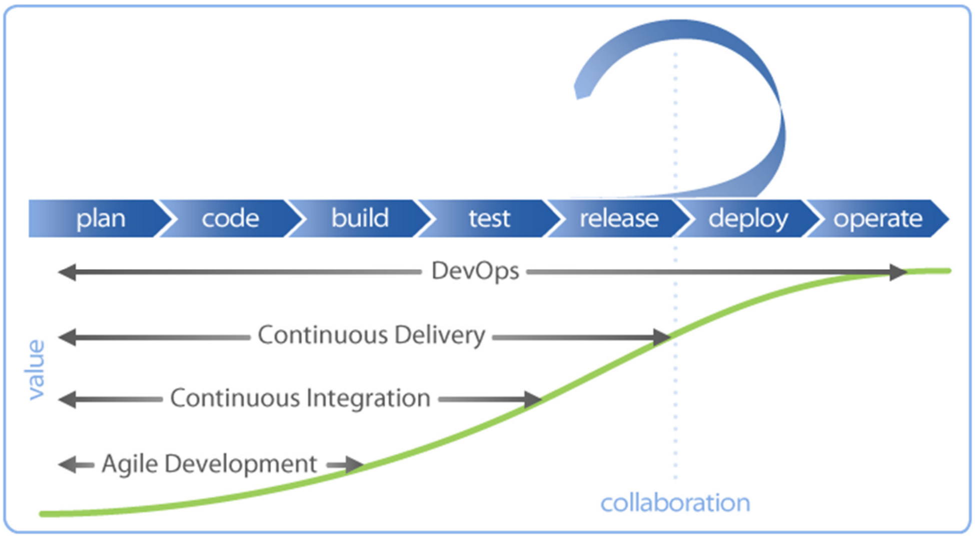


Figure 2 DevOps vs Continuous Delivery

Some of the phases may not fit in all kinds of services or products. Ex. ‘Operation’ phase is not relevant in the case of a product that is shipped to an external customer. The operation of that product falls under the customer’s operation life cycle.

2.2.DevOps Goals

The goals of DevOps are to make improvements across all components in the product and service delivery. They include

- Improved deployment frequency
- Faster time to market
- Lower failure rates of new releases
- Faster recovery time from crashes or failures.

2.3.DevOps Benefits

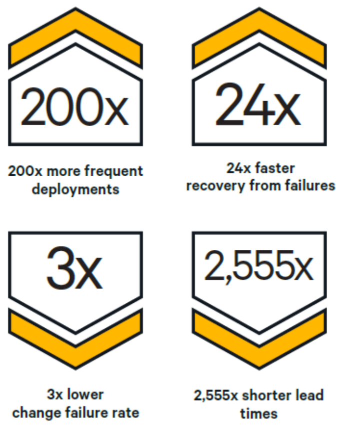
Companies that practice DevOps have reported significant benefits. Some of them are

- Significantly shorter time-to-market
- Improved customer satisfaction
- Better product quality
- More reliable releases
- Improved productivity and efficiency
- The increased ability to build the right product by fast experimentation

Some of the key findings from PuppetLabs/DORA study in 2016 are as follows.

High-performing IT organizations compared to low-performing organizations

- 200x more frequent deployments
- 24x faster recovery times
- 2,555 times faster lead times
- 2.2x better employee loyalty (eNPS)
- 22% less time on unplanned work and rework
- 50 percent less time remediating security issues


200x
200x more frequent deployments
24x
24x faster recovery from failures
3x
3x lower change failure rate
2,555x
2,555x shorter lead times

2.4.DevOps Practices

DevOps is more than just a set of practices. There are many that are used in the industry. The following section will list the top and commonly used practice in a DevOps process in an IT organization.

i. Start small.
Trying to do too much at once is a recipe for disaster. It is better to start small projects and gain confidence of team. Start project that has a high success probability. This could be a pilot or prototype or proof of concept project.

ii. Concentrate on the process and not on the tools.
It is better to be tool independent and a good process should have the ability to replace one tool with another without too much disruption.

iii. Put everything under version control.
To reliably deploy an application on to production, all the items that are required for building the application: code, test cases, design documents, external libraries, databases and anything that could be updated, has to be put under version control.

iv. Maintain a production-equivalent staging environment
Typically a development environment is different from production. So, to avoid issues that are found only after going into production, like performance, access related and such, it is advised to have a staging environment where all those issues can be found before production.

v. Frequent deployment to production
To make any new feature, bug fixes, security patches to the customer ASAP, it is suggested to deploy code frequently. It could be once for each sprint cycle in an Agile development setup, or once every duration as fixed by the DevOps team (part of Release Management).

vi. Automate build of app environment
For a simple application, this is not much of a concern. But for most complex applications, setting up the run environment, like setting load balancer, databases, web server, network interconnectivity and all things operation oriented, it is better to have an automated way to

build environment to reduce time to setup and also reduce any human errors. Infrastructure as a Service (IAAS) is used for this purpose.

vii. Automate deployment of application
In addition to above environment build automation, it is also very beneficial to have automation to deploy the application to the environment. This includes initializing the databases, initial setup of application, installation of dependencies and others. The automated build and deployment can be achieved through 'Infrastructure as a code' methodology

2.5.DevOps Tools

Since DevOps is more of a process, there is not a single tool that helps in deploying DevOps practice in an organization. It is more of a 'toolchain', a suite of tools that helps in the implementation of a chosen DevOps practice. DevOps implementation doesn't mean to throw away existing tools in favour of new ones. It involves in integrating the existing tools and migrating the non-compliant tools to fit the chosen practice/process. The process steps you need to consider for tools support are:

- Request capture and ticket workflow
- Source Control
- Agile Planning
- Test Case Management
- Build automation
- Continuous deployment
- Release Management
- Automated test scripts and Load Testing
- Feedback Management
- Team Collaboration
- Application Telemetry
- Lab Management
- Cloud provider integration

Now there are many tools that have become platforms through which many functions can be performed with a single tool. Some of the popular DevOps platforms are:

Jenkins

Jenkins is the leading open source, extendable continuous integration server on the market today. Written in Java, Jenkins provides automated continuous integration services for software development. The platform provides considerable flexibility around how builds can be initiated – including a version control system, scheduled cron jobs, kicked off when other builds have completed, and by means of a specific build URL.

Jenkins is also ubiquitous in the DevOps market, supported by well over 800 plugins that allow users to extend its use to support projects written in almost any language, most version control systems, and most large databases.

Chef offers what it calls Automation for Web-Scale IT, by delivering fast, scalable, flexible IT automation. Chef is a configuration management platform that makes use of what it calls “recipes” to automate infrastructure tasks. Examples of recipes are instructions for configuring web servers, databases and load balancers. These recipes describe what an infrastructure consists of and how each part of it should be deployed, configured and managed.

Chef can streamline the configuration and maintenance of a company’s servers, as well as integrate with cloud-based platforms such as Rackspace, Amazon EC2, and Google Cloud Platform to automatically provision and configure new machines.

Puppet Labs is a popular name in the DevOps industry, and is known for its open-source configuration management tool, Puppet, as well as the commercially available version, known as Puppet Enterprise. Puppet Enterprise is used to automate tasks at different stages of the IT infrastructure lifecycle, including discovery, provisioning, OS and application configuration management, orchestration, and reporting.

Puppet Labs offers an expansive ecosystem that includes training and certification. Through Puppet Certification Program Puppet Labs provides professional exams to recognize IT professionals who have demonstrated the technical know-how and experience needed to manage their infrastructure proactively throughout its lifecycle.

AnsibleWorks provides IT orchestration engine that makes applications and systems easier to deploy. Or in its own words, “Ansible makes deploying software fun again.” Ansible takes this claim seriously. Unlike most configuration management platforms, Ansible uses what is called “agentless architecture.” In most cases, nodes must have a locally installed daemon that communicates with a controlling machine. Ansible, on the other hand, doesn’t require a custom agent or software to install, but simply communicates over SSH.

Ansible is easily extensible and can deploy to virtualization environments and public and private cloud environments including VMware, OpenStack, AWS, Eucalyptus Cloud, KVM, and CloudStack. The platform also can deploy big data, storage and analytics environments including Hadoop, Riak, and Aerospike.

3. Riverstone

Riverstone was started in the U.S. in the year 2000 and in India in the year 2001. Riverstone is a company with its culture and focus firmly on its own team. We believe that a team-centric company can learn and grow together and ultimately meet the requirements of our customers with quality better than anyone else can. We take pride in the capabilities of our team, encourage each and every member of the team to go above and beyond the call of duty, reward the team for the hard work and constantly foster a culture of discipline, learning, growth and above all, quality.

At Riverstone we deem in delivering high quality & reliable solutions to complex business problem of clients globally with our innovative & highly professional methodologies. With our continued dedication we look to become a "single stop shop" for all the IT needs of our customers. Over the past 12 years, our mission is fully focused to become the Next Generation IT services provider. With more and more small/medium enterprises looking to outsource their software development and business processes, the rules are rapidly changing. Our Professional Services team is very experienced and can deliver high quality and competitive DevOps and IT automation solutions to our customers.

4. DevOps Implementation

Implementing a good DevOps process will benefit any organization to a great extent. This includes planning, developing, integration and automation. There are a few product areas that can immediately benefit from some integration and automation tasks that will reduce the manual work involved, increase the responsiveness of support team and also optimize the resources used. A typical DevOps implementation will integrate and automate the different process with tools appropriate to the customer and would look like the image below. (PC: digitalcto.com)

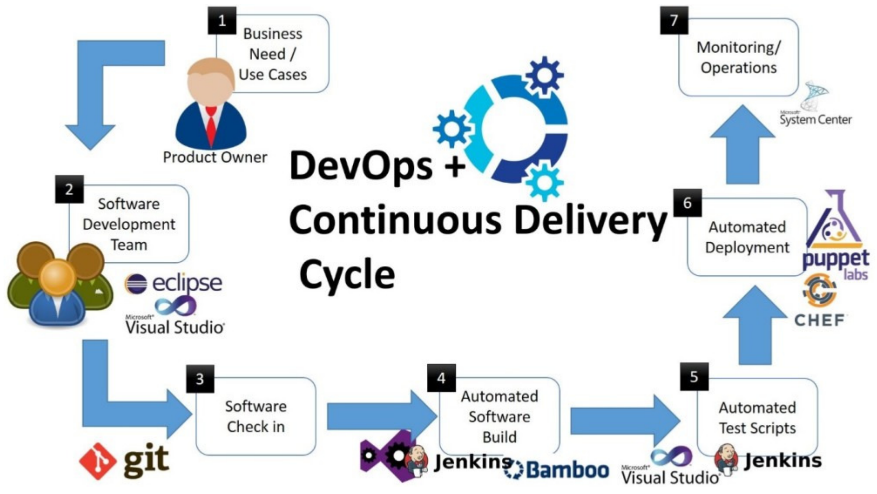


Figure 3 DevOps process and tools

The Riverstone Professional Services personnel will engage with customer and listed below are the steps that Riverstone will do as part of their service.

- Understand the current product and service lifecycle from requirement to production (and maintenance)

- Finalize with the management the objectives for a DevOps implementation
- Document (if not existing earlier) existing processes and systems
- Create a baseline measurement of the current DevOps tools and capability
- Uncover opportunities for improvement
- Identify the most impactful areas to the business
- Document ideal end-state for teams
- Generate and present a roadmap to implement/improve DevOps process using relevant platform and tools
- Prototype implementation to improve one area of improvement.

4.1.Understanding current situation

Riverstone will engage in conversation with the respective teams in the organization and asses the current DevOps capabilities and the tools used. This is done with the help of a questionnaire that will be shared with the stakeholders. A properly filled up questionnaire will give a good understanding of the current setup at the customer organization.

4.2.Finalizing the DevOps objectives

Riverstone with the help of stakeholders will finalize the DevOps objectives of the implementation and also an appropriate timeline for achieving those objectives.

4.3.Documentation of existing processes

The current processes, tools & systems used by the customer will be documented, if no such document exists already with the customer.

4.4.Baseline measurement

Before starting the project, some measurements to indicate the current DevOps tools and capability will be noted down. Some examples of these measurements would be: Number of releases in a year; Average time to fix a defect and so on. This measurement will be compared against after the engagement is completed.

4.5.Opportunities for improvement

During the analysis of the current process and tools, there will be some areas that will require improvement in terms of process or technology. This will be incorporated into the final presentation.

4.6.Impactful areas

Each area of improvement will be analyzed for business impact and the areas with high impact will be documented.

4.7.Ideal end-state

After all the analysis are done and processed, the ideal end-state from a DevOps standpoint will be documented and presented. This will include the process and tools that will fit the customer requirement. Some of the process would be: Code review process; Versioning best practice; Infrastructure recommendation to use of private cloud; and so on.

4.8.Roadmap

Finally, a roadmap that will show the steps to implement a new DevOps process or to improve the existing process and the list of tools that are recommended will be presented to the customer.

4.9.Prototype

Riverstone would then show the benefit of new DevOps process by prototyping and demonstrating one of the areas identified for improvement.

5. Use Case Scenarios

Listed below are some of the scenarios that Riverstone will help the customer implement the DevOps process

5.1.Integration with Continuous Integration (CI) platform

CI is an important component of the DevOps process. Tools like Jenkins, Buildbot, Microsoft VSTS/TFS, and Bamboo are the popular CI tools. One of them can be implemented based on customer requirements.

CI tools can be integrated with supported version control system like Git, Subversion to provide automated builds whenever there is a code commit in any chosen branch. This avoids a manual way to someone checking out code and doing a build. Any build errors because of a wrong commit can also be caught immediately rather than later during a periodic builds.

CI can be further extended to take such builds and run automated test cases which on any failed test cases can automatically revert the committed code. Tools like Jenkins provide a lot of plugins which helps in extending Jenkins to do many functions within a DevOps process.

Take the below diagram as a typical development process. The developer commits the code into a repository system (ex. Git). The build and release team part of the Operations will take the committed code and build it on the build server. Then they deploy the resultant bits (like War file) on to a testing environment. The test/QA team then does their tests on the application. On successful testing, the release team then deploys the application on to the production server.

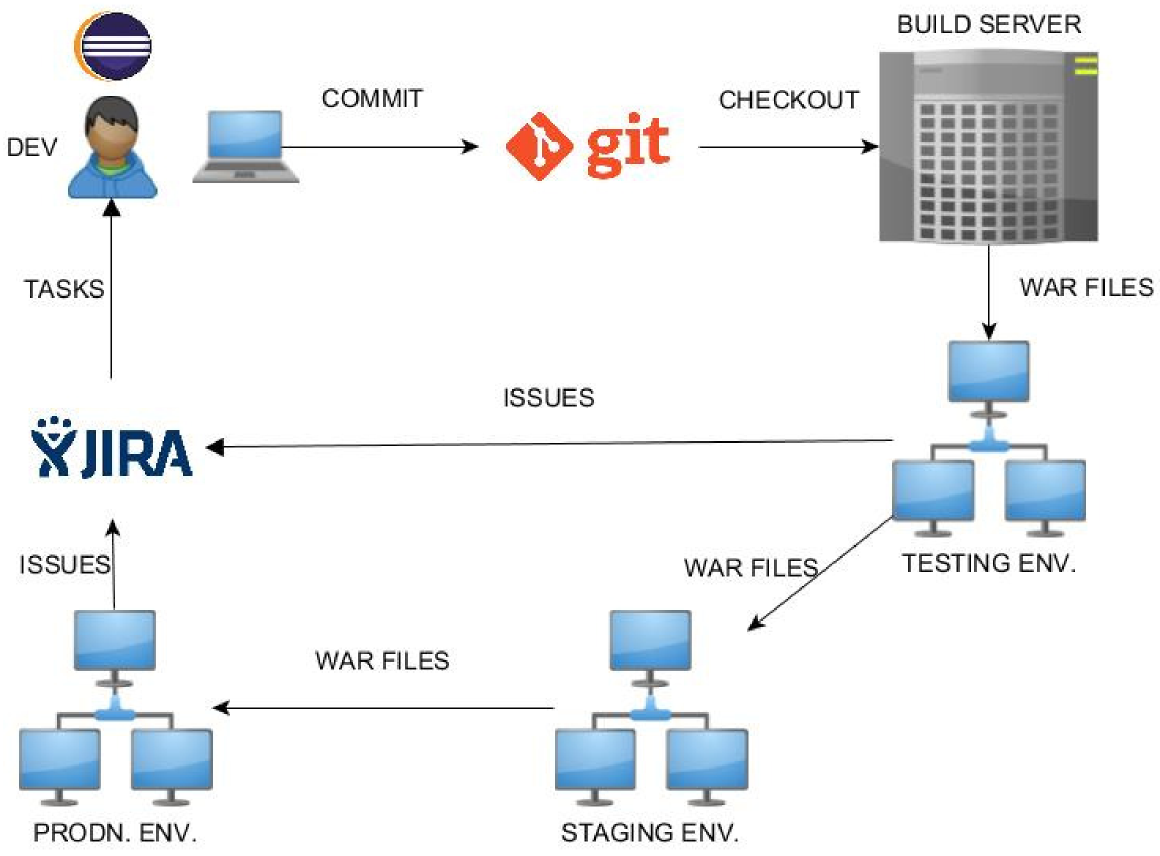


Figure 4 Manual processes in a development life cycle

There are many manual steps in such a process.

i. Checking out code to build server
ii. Doing the build
iii. Deploying the application to test server
iv. Testing by the QA team (manual part, if not fully automated)
v. Deploying the application to production server
vi. Deployment of build server
vii. Deployment of test server
viii. Monitoring of the test, staging, production servers

Now we install a CI tool (ex. Jenkins) and integrate it with the version control system (Git), testing server and also with the staging and production setup. With proper configuration, the CI server will automatically checkout code from the Git repository (can be configured to do daily/weekly or whenever a commit happens). It then uses the build servers to build the application. It can deploy the application to the configured test server and run the automated test cases. On successful test, it can deploy the application to staging environment and eventually on to production environment on request. This process now eliminates a lot of manual steps. The below diagram denotes the new process.

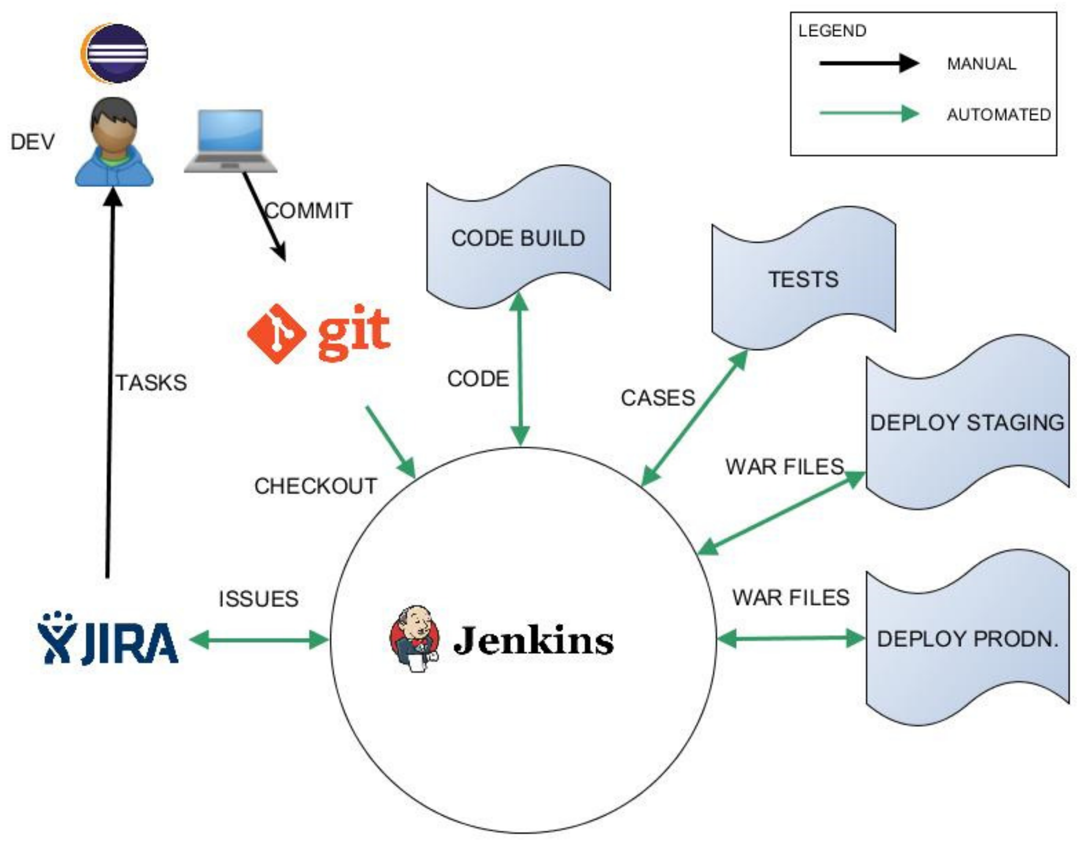


Figure 5 Integrating with a CI tool

The new automated process will eliminate human errors if any, freeing up human resources who can be deployed to more productive work, increases the rate of releases.

5.2.Integration with Infrastructure/Configuration Management

Continuous Deployment is another major component of the process. To do application deployments automatically, the infrastructure (CPU, storage, memory) should support it. This is possible when customer uses an IAAS platform. It could be a private/on premise cloud like VMware, OpenStack or public/internet cloud like Amazon AWS, Microsoft Azure.

CI build setup, testing setup and many other functions requires hosts with certain CPU/memory/storage capacity. On this the builds would happen, applications would be deployed for system testing and so on. Instead of setting up dedicated build servers or testing servers which will be used only when required and will be idle for the reminder of the time, it is better to deploy these hosts as and when it is required. This is possible with technologies such as Infrastructure as a Service (IAAS), configuration management. CI tools like Jenkins and others have well integrated interfaces to IAAS platforms like OpenStack, AWS which can provision the required build servers on demand and do the build for it. Similarly, testing automation can use the appropriate interfaces (using APIs) to provision the required machines (virtual machines) and use configuration management tools such as Chef, Puppet to deploy the applications on to these VMs and then do the automated testing. Most of these will be done automatically in a DevOps process.

Take the example mentioned in 4.1. Even after automating the build and release process using a CI (Jenkins) platform, there are some manual steps involved. The Operations team will be involved in setting up the hosts for build servers, for deploying the applications into testing and staging environments. They also need to install the dependent software like databases, directory servers and so on. These are all very much manual process. Not only that but the deployed hosts will always be up and running even when no build or testing takes place. This is a waste of company resources. The whole process would look like the picture below.

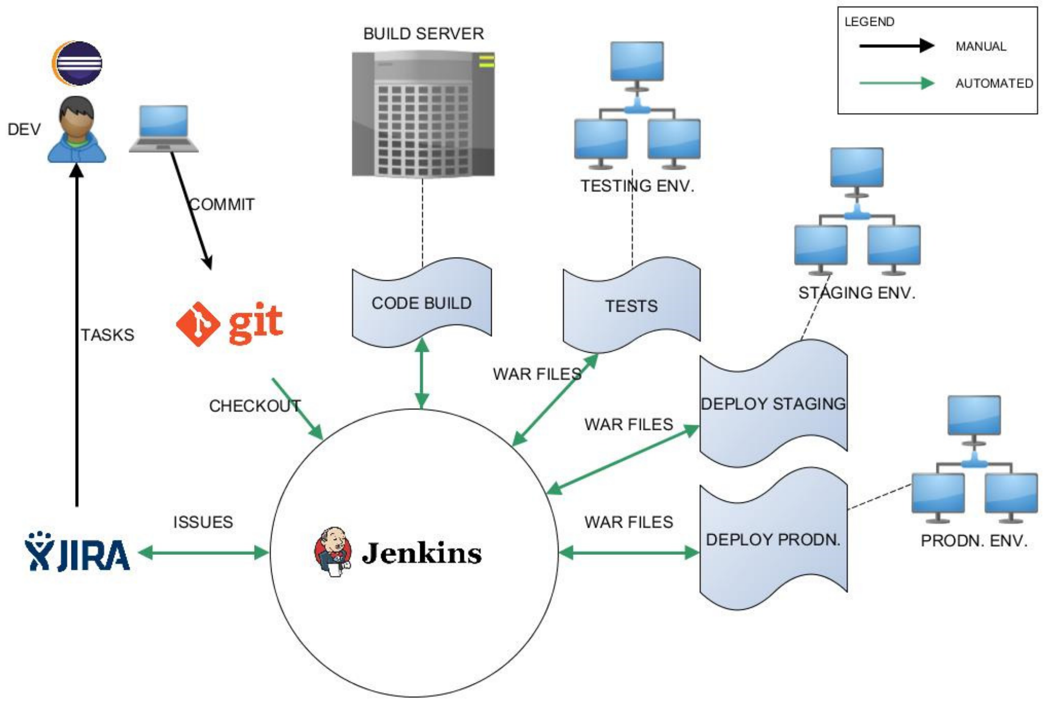


Figure 6 Manual provisioning of build/test hosts

The use of configuration management tools such as Chef along with IAAS platform like OpenStack can automate the manual process. This is usually done by requesting hosts (as virtual machines) and then using Chef recipes to deploy the application. In some cases the CI tool like Jenkins can do the application deployment by itself and in those cases tools like Chef may not be required. The new process would then look like below.

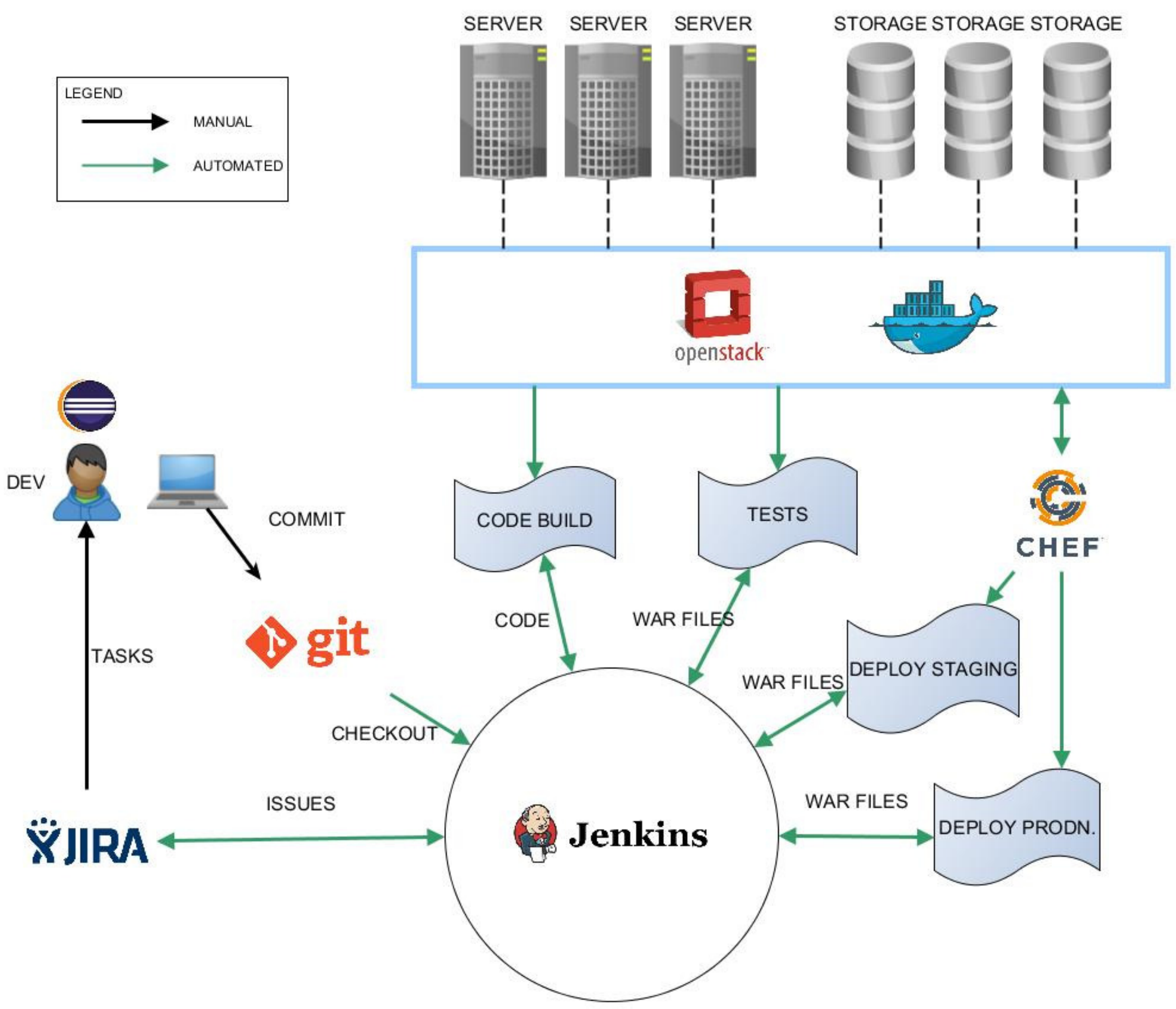


Figure 7 Integration with IAAS & CM

The new process will remove manual deployment of hosts and dependent software installation. The build servers and test servers are deployed only when required there by using the resources efficiently.

5.3.Integration with Monitoring solution

Continuous monitoring is yet another important component of the DevOps process. Monitoring is done for all the categories in an environment: Performance, Capacity, Throughput, Uptimes, KPIs, Compliance, Log files. Each of the categories will have different consumers like Developers, Testers, IT sysadmins and so on.

With a good setup the monitoring can be integrated with other components of the process. For example, whenever there is a capacity issue, like storage running short, the Operations team could be notified of such an event. If integrated with IAAS setup, this could be automated to provide more disk capacity on request. Similarly, whenever there is a performance issue with the deployed application, the process can be used to automatically raise an incident with appropriate logs which can be looked into by the development team to be fixed in the next iteration. The process will look like below after adding monitoring to the process.

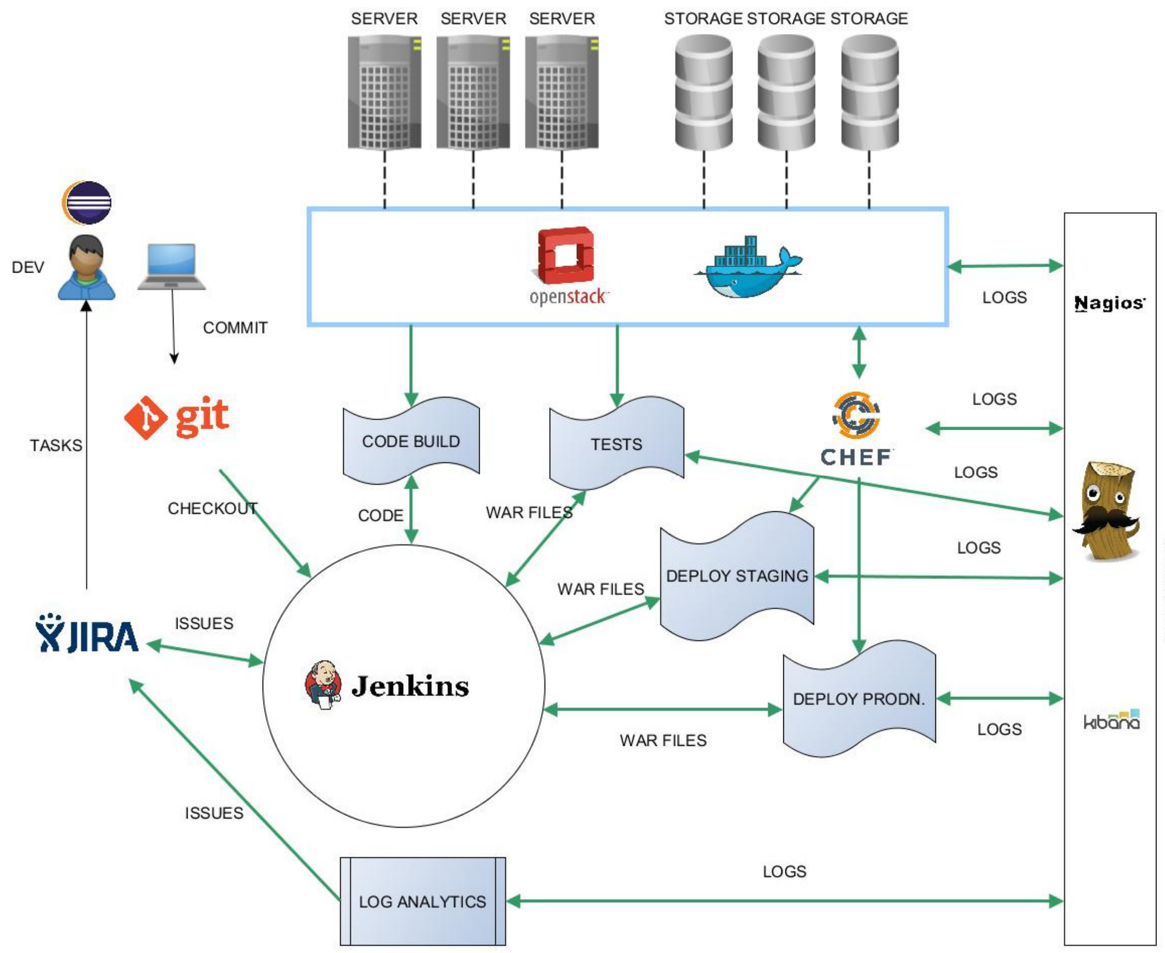


Figure 8 Integration with Monitoring

The monitoring is usually captured by logging. There are tools like Nagios, ELK (Electricsearch, Logstash, Kibana) stack that can be deployed to do logging and monitoring. The captured logs are then analyzed to find any issues and then added to the request tracker for further processing.