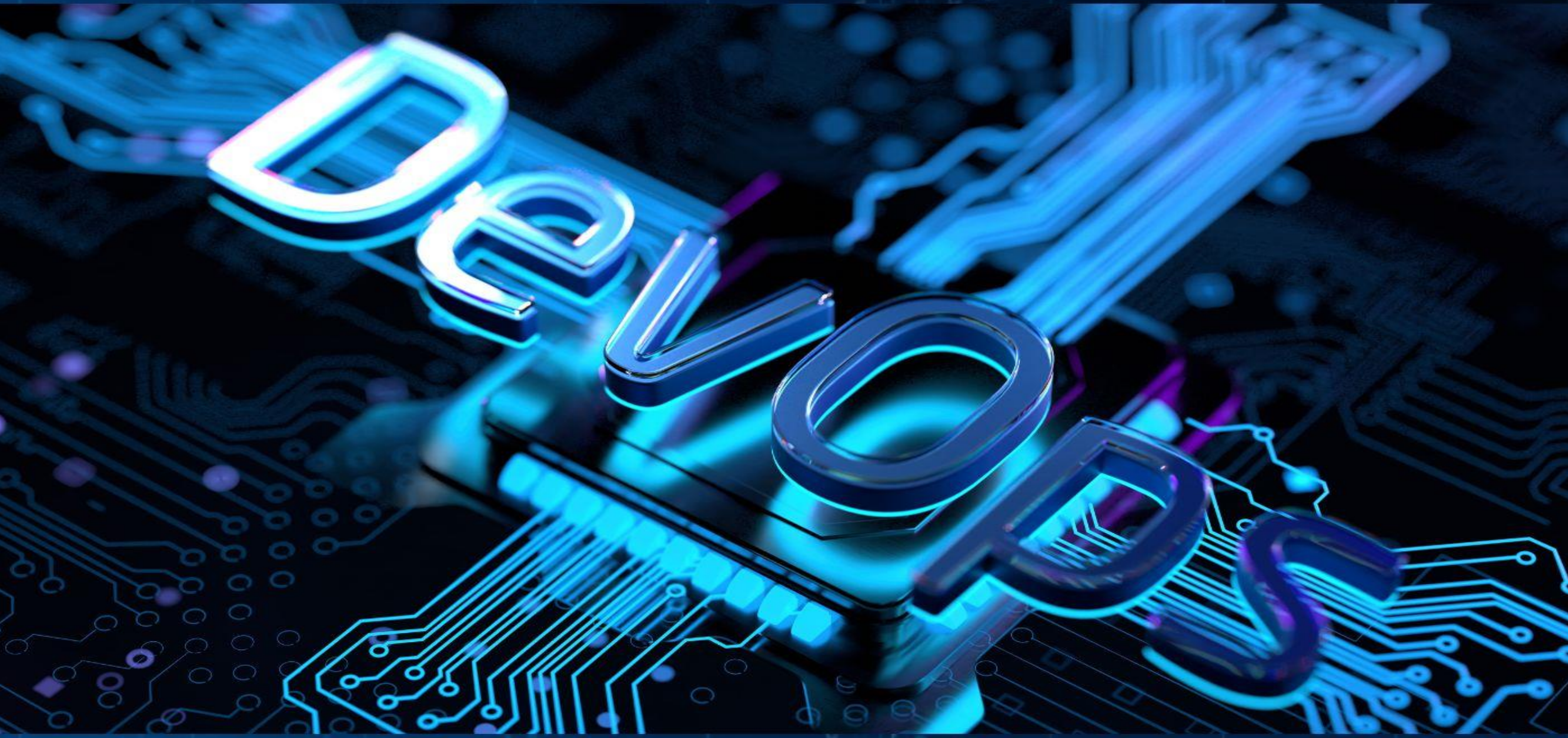
SADIQ TECH SOLUTIONS

DevOps Kubernetes E-BOOK



kubernetes

https://topitcourses.com

Table of Contents

Preface

This book walks you through a journey of learning fundamental concept and useful skills for DevOps, containers and Kubernetes.

What this book covers

`Chapter 1`, *Introduction to DevOps*, walks you through the evolution from the past to what we call DevOps today and the tools that you should know. Demand for people with DevOps skills has been growing rapidly over the last few years. It has accelerated software development and delivery speed and has also helped business agility.

`Chapter 2`, *DevOps with Container*, helps you learn the fundamentals and container orchestration. With the trend of microservices, container has been a handy and essential tool for every DevOps because of its language agnostic isolation.

`Chapter 3`, *Getting Started with Kubernetes*, explores the key components and API objects in Kubernetes and how to deploy and manage containers in a Kubernetes cluster. Kubernetes eases the pain of container orchestration with a lot of killer features, such as container scaling, mounting storage systems, and service discovery.

`Chapter 4`, *Working with Storage and Resources*, describes volume management and also explains CPU and memory management in Kubernetes. Container storage management can be hard in a cluster.

`Chapter 5`, *Network and Security*, explains how to allow inbound connection to access Kubernetes services and how default networking works in Kubernetes. External access to our services is necessary for business needs.

`Chapter 6`, *Monitoring and Logging*, shows you how to monitor a resource's usage at application, container, and node level using Prometheus. This chapter also shows how to collect logs from your applications, as well as Kubernetes with Elasticsearch, Fluentd, and Kibana stack. Ensuring a service is up and healthy is one of the major responsibilities of DevOps.

`Chapter 7`, *Continuous Delivery*, explains how to build a Continuous Delivery pipeline with GitHub/DockerHub/TravisCI. It also explains how to manage updates, eliminate the potential impact when doing rolling updates, and prevent possible failure. Continuous Delivery is an approach to speed up your time-to-market.

`Chapter 8`, *Cluster Administration*, describes how to solve the preceding problems with the Kubernetes namespace and ResourceQuota and how to do access control in Kubernetes. Setting up administrative boundaries and access control to Kubernetes cluster are crucial to DevOps.

`Chapter 9`, *Kubernetes on AWS*, explains AWS components and shows how to provision Kubernetes on AWS. AWS is the most popular public cloud. It brings the infrastructure agility and flexibility to our world.

`Chapter 10`, *Kubernetes on GCP*, helps you understand the difference between GCP and AWS, and the benefit of running containerized applications in hosted service from Kubernetes' perspective. Google Container Engine in GCP is a managed environment for Kubernetes.

`Chapter 11`, *What's Next?*, introduces other similar technologies, such as Docker Swarm mode, Amazon ECS, and Apache Mesos and you'll have an understanding of which the best approach is for your business. Kubernetes is open. This chapter will teach you how to get in touch with Kubernetes community to learn ideas from others.

What you need for this book

This book will guide you through the methodology of software development and delivery with Docker container and Kubernetes using macOS and public cloud (AWS and GCP). You will need to install minikube, AWSCLI, and the Cloud SDK to run the code samples present in this book.

Who this book is for

This book is intended for DevOps professionals with some software development experience who are willing to scale, automate, and shorten software delivery to the market.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Mount the downloaded `WebStorm-10*.dmg` disk image file as another disk in your system."

Any command-line input or output is written as follows:

```
$ sudo yum -y -q install nginx
$ sudo /etc/init.d/nginx start
Starting nginx:
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "The shortcuts in this book are based on the **Mac OS X 10.5+** scheme."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book-what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

Downloading the example code

You can download the example code files for this book from your account at `https://.topitcourses.com`. If you purchased this book elsewhere, you can visit `http://www.topitcourses.com/support` and register to have the files emailed directly to you. You can download the code files by following these steps:

1. Browser the above mentioned url.
2. Pay the fee and download the pdf file.
3. Click on **Download**.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

TopITCourses.com (sometimes styled *Top IT Courses*) is a platform that offers **IT-related learning materials, primarily e-books and downloadable tech guides** covering topics such as:

- Data Science with AI
- Core Java
- Python Full Stack Development
- Web Development, React JS
- DevOps/Kubernetes
- OKTA IAM
- UI/UX & Digital Marketing
  …and other IT skill guides.

What You Can Find on the Site

- IT course **e-books for download** (often priced individually).
- Topics include programming languages, design, cybersecurity, and more.
- Customer support contact and transaction pages for buying products.
- The site also lists refund, shipping, and policy information.
- Promotional posts on social platforms (e.g., Instagram) advertising specific e-books like AI guides.

Takeaway

TopITCourses.com operates an online portal where you can **buy downloadable IT learning e-books** on a range of tech topics. However:

- There's **no widely recognized accreditation** or verification of the platform's educational quality.
- It's best to compare these resources with well-known online learning platforms (like Coursera, edX, Udemy) before deciding to pay for content.

If you'd like, I can help you check **reviews or legitimacy details** further or suggest **trusted alternatives for learning IT skills online**!

Questions

If you have a problem with any aspect of this book, you can contact us at `info@topitcourses.com`, and we will do our best to address the problem.

1
Introduction to DevOps

Software delivery cycle has been getting shorter and shorter, while on the other hand, application size has been getting bigger and bigger. Software developers and IT operators are under the pressure to find a solution to this. There is a new role, called **DevOps**, which is dedicated to support software building and delivery.

This chapter covers the following topics:

- How has software delivery methodology changed?
- What is microservice, and why do people adopt this architecture?
- How does DevOps support to build and deliver the application to the user?

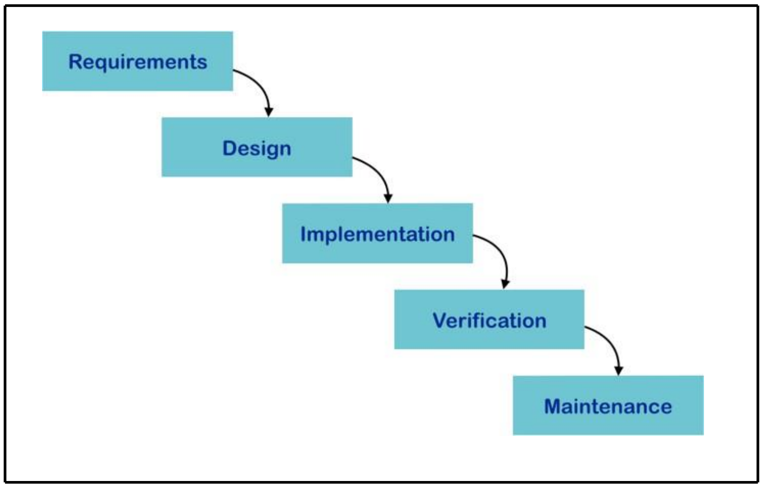
Software delivery challenges

Building a computer application and delivering it to the customer has been discussed and has evolved over time. It is related to **Software Development Life Cycle** (**SDLC**); there are several types of processes, methodologies, and histories. In this section, we will describe its evolution.

Waterfall and physical delivery

Back in the 1990s, software delivery was adopted by a **physical** method, such as a floppy disk or a CD-ROM. Therefore, SDLC was a very long-term schedule, because it was not easy to (re)deliver to the customer.

At that moment, a major software development methodology was a **waterfall model**, which has requirements/design/implementation/verification/maintenance phases as shown in the following diagram:



In this case, we can't go back to the previous phase. For example, after starting or finishing the **Implementation** phase, it is not acceptable to go back to the **Design** phase (to find a technical expandability issue, for example). This is because it will impact the overall schedule and cost. The project tends to proceed and complete to release, then it goes to the next release cycle including a new design.

It perfectly matches the physical software delivery because it needs to coordinate with logistics management that press and deliver the floppy/CD-ROM to the user. Waterfall model and physical delivery used to take a year to several years.

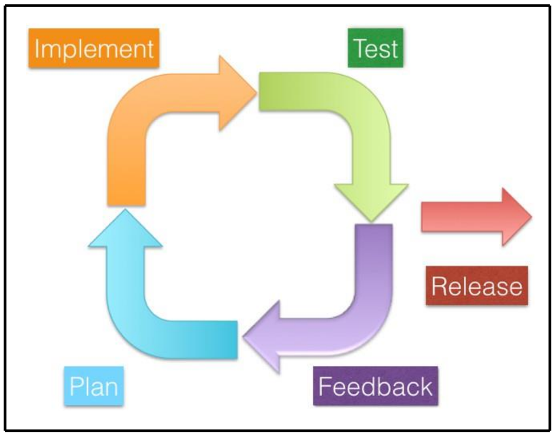
Agile and electrical delivery

A few years later, the internet became widely accepted, and then software delivery method also changed from physical to **electrical**, such as online download. Therefore, many software companies (also known as dot-com companies) tried to figure out how to shorten the SDLC process in order to deliver the software that can beat the competitors.

Many developers started to adopt new methodologies such as incremental, iterative, or **agile** models and then deliver to the customer faster. Even if new bugs are found, it is now easier to update and deliver to the customer as a patch by electrical delivery. Microsoft Windows update was also introduced since Windows 98.

In this case, the software developer writes only a small logic or module, instead of the entire application in one shot. Then, it delivers to the QA, and then the developer continues to add a new module and finally delivers it to the QA again.

When the desired modules or functions are ready, it will be released as shown in the following diagram:



This model makes the SDLC cycle and the software delivery faster and also easy to be adjust during the process, because the cycle is from a few weeks to a few months which is small enough to make a quick adjustment.

Although this model is currently favoured by the majority, at that moment, application software delivery meant software binary, such as EXE program which is designed to be installed and run on the customer's PC. On the other hand, the infrastructure (such as server and network) is very static and set up beforehand. Therefore, SDLC doesn't tend to include these infrastructures in the scope yet.

Software delivery on the cloud

A few years later, smartphones (such as iPhone) and wireless technology (such as Wi-Fi and 4G network) became widely accepted, and software application also changed from binary to the online service. The web browser is the interface of the application software, which need not be installed anymore. On the other hand, infrastructure becomes dynamic, since the application requirement keeps changing and the capacity needs to grow as well.

Virtualization technology and **Software Defined Network** (**SDN**) make the server machine dynamic. Now, cloud services such as **Amazon Web Services** (**AWS**) and **Google Cloud Platform** (**GCP**) can be easy to create and manage dynamic infrastructures.

Now, infrastructure is one of the important components and being within a scope of Software Development Delivery Cycle, because the application is installed and runs on the server side, rather than a client PC. Therefore, software and service delivery cycle takes between a few days to a few weeks.

Continuous Integration

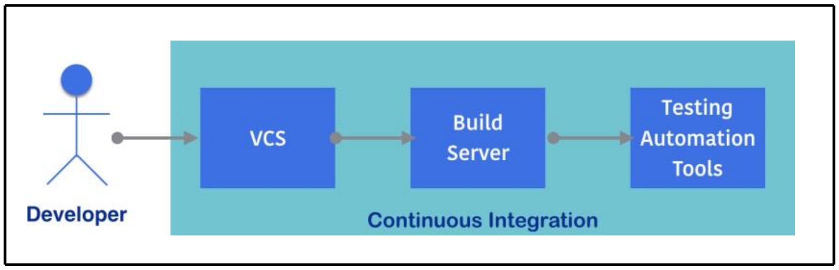
As discussed previously, the surrounding software delivery environment keeps changing; however, the delivery cycle is getting shorter and shorter. In order to achieve rapid delivery with higher quality, the developer and QA start to adopt some automation technologies. One of the popular automation technologies is **Continuous Integration** (**CI**). CI contains some combination of tools, such as **Version Control Systems** (**VCS**), **build server**, and **testing automation tools**.

VCS helps the developer to maintain program source code onto the central server. It prevents overwriting or conflict with other developers' code and also preserves the history. Therefore, it makes it easier to keep the source code consistent and deliver to the next cycle.

The same as VCS, there is a centralized build servers that connects VCS to retrieve the source code periodically or automatically when the developer updates the code to VCS, and then trigger a new build. If the build fails, it notifies the developer in a timely manner. Therefore, it helps the developer when someone commits the broken code into the VCS.

Testing automation tools are also integrated with build server that invoke the unit test program after the build succeeds, then notifies the result to the developer and QA. It helps to identify when somebody writes a buggy code and stores to VCS.

The entire flow of CI is as shown in the following diagram:



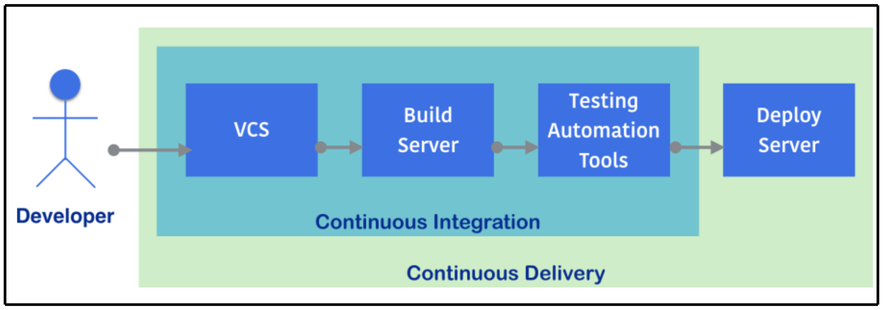
CI helps both the developer and the QA not only to increase the quality, but also to shorten archiving an application or module package cycle. In an age of electrical delivery to the customer, CI is more than enough. However, because delivery to the customer means to deploy to the server.

Continuous Delivery

CI plus deployment automation is the ideal process for the server application to provide a service to customers. However, there are some technical challenges that need to be resolved. How to deliver a software to the server? How to gracefully shutdown the existing application? How to replace and rollback the application? How to upgrade or replace if the system library also needs to change? How to modify the user and group settings in OS if needed? and so on.

Because the infrastructure includes the server and network, it all depends on an environment such as Dev/QA/staging/production. Each environment has different server configuration and IP address.

Continuous Delivery (**CD**) is a practice that could be achieved; it is a combination of CI tool, configuration management tool, and orchestration tool:



Configuration management

The configuration management tool helps to configure an OS including the user, group, and system libraries, and also manages multiple servers that keep consistent with the desired state or configuration if we replace the server.

It is not a scripting language, because scripting language performs to execute a command based on the script line by line. If we execute the script twice, it may cause some error, for example, attempt to create the same user twice. On the other hand, configuration management looks at the **state**, so if user is created already, the configuration management tool doesn't do anything. But if we delete a user accidentally or intentionally, the configuration management tool will create the user again.

It also supports to deploy or install your application to the server. Because if you tell the configuration management tool to download your application, then set it up and run the application, it tries to do so.

In addition, if you tell the configuration management tool to shut down your application, then download and replace to a new package if available, and then restart the application, it keeps up to date with the latest version.

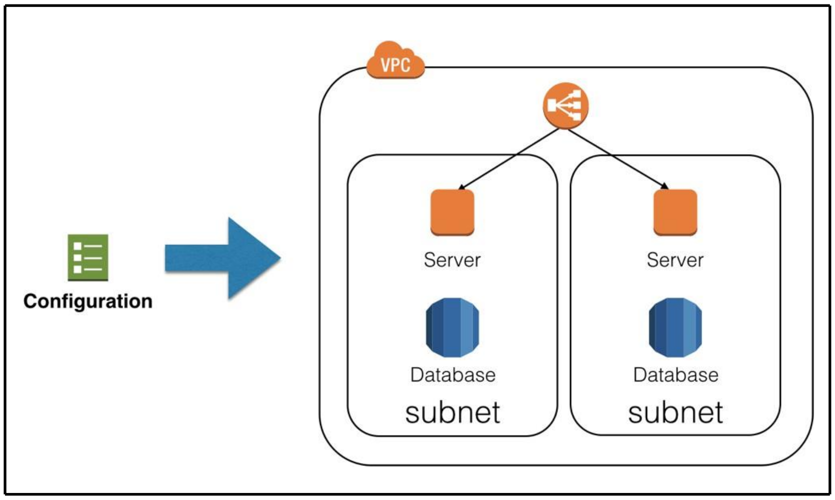
Of course, some of the users want to update the application only when it is required, such as blue-green deployments. The configuration management tool allows you to trigger to execute manually too.

Blue-green deployments is a technique that prepares the two sets of application stack, then only one environment (example: blue) is servicing to the production. Then when you need to deploy a new version of application, deploy to the other side (example: green) then perform the final test. Then if it works fine, change the load balancer or router setting to switch the network flow from blue to green. Then green becomes a production, while blue becomes dormant and waiting for the next version deployment.

Infrastructure as code

The configuration management tool supports not only OS or Virtual Machine, but also cloud infrastructure. If you need to create and configure a network, storage, and Virtual Machine on the cloud, it requires some of the cloud operations.

But the configuration management tool helps to automate the setup cloud infrastructure by configuration file, as shown in the following diagram:



Configuration management has some advantage against maintaining an operation manual **Standard Operation Procedure** (**SOP**). For example, maintaining a configuration file using **VCS** such as Git, you can trace the history of how the environment setting has changed.

It is also easy to duplicate the environment. For example, you need an additional environment on cloud. If you follow the traditional approach, (that is, to read the SOP document to operate the cloud), it always has a potential human error and operation error. On the other hand, we can execute the configuration management tool that creates an environment on cloud quickly and automatically.

Infrastructure as code may or may not be included in the CD process, because infrastructure replacement or update cost is higher than just replacing an application binary on the server.

Orchestration

The orchestration tool is also categorized as one of the configuration management tools. However its more intelligent and dynamic when configuring and allocating the cloud resources. For example, orchestration tool manages several server resources and networks, and then when the administrator wants to increase the application instances, orchestration tool can determine an available server and then deploy and configure the application and network automatically.

Although orchestration tool is beyond the SDLC, it helps Continuous Delivery when it needs to scale the application and refactor the infrastructure resource.

Overall, the SDLC has been evolved to achieve rapid delivery by several processes, tools, and methodologies. Eventually, software (service) delivery takes anywhere from a few hours to a day. While at the same time, software architecture and design has also evolved to achieve large and rich applications.

Trend of microservices

Software architecture and design also keep evolving, based on the target environment and volume of the application's size.

Modular programming

When the application size is getting bigger, developers tried to divide by several modules. Each module should be independent and reusable, and should be maintained by different developer teams. Then, when we start to implement an application, the application just initializes and uses these modules to build a larger application efficiently.

The following example shows what kind of library Nginx (`https://www.nginx.com`) uses on CentOS 7. It indicates that Nginx uses `OpenSSL`, `POSIX thread` library, `PCRE` the regular expression library, `zlib` the compression library, `GNU C` library, and so on. So, Nginx didn't reinvent to implement SSL encryption, regular expression, and so on:

```
$ /usr/bin/ldd /usr/sbin/nginx
  linux-vdso.so.1 =>  (0x00007ffd96d79000)
  libdl.so.2 => /lib64/libdl.so.2 (0x00007fd96d61c000)
  libpthread.so.0 => /lib64/libpthread.so.0
  (0x00007fd96d400000)
  libcrypt.so.1 => /lib64/libcrypt.so.1
  (0x00007fd96d1c8000)
  libpcre.so.1 => /lib64/libpcre.so.1 (0x00007fd96cf67000)
  libssl.so.10 => /lib64/libssl.so.10 (0x00007fd96ccf9000)
  libcrypto.so.10 => /lib64/libcrypto.so.10
  (0x00007fd96c90e000)
  libz.so.1 => /lib64/libz.so.1 (0x00007fd96c6f8000)
  libprofiler.so.0 => /lib64/libprofiler.so.0
  (0x00007fd96c4e4000)
   libc.so.6 => /lib64/libc.so.6 (0x00007fd96c122000)
   ...
```

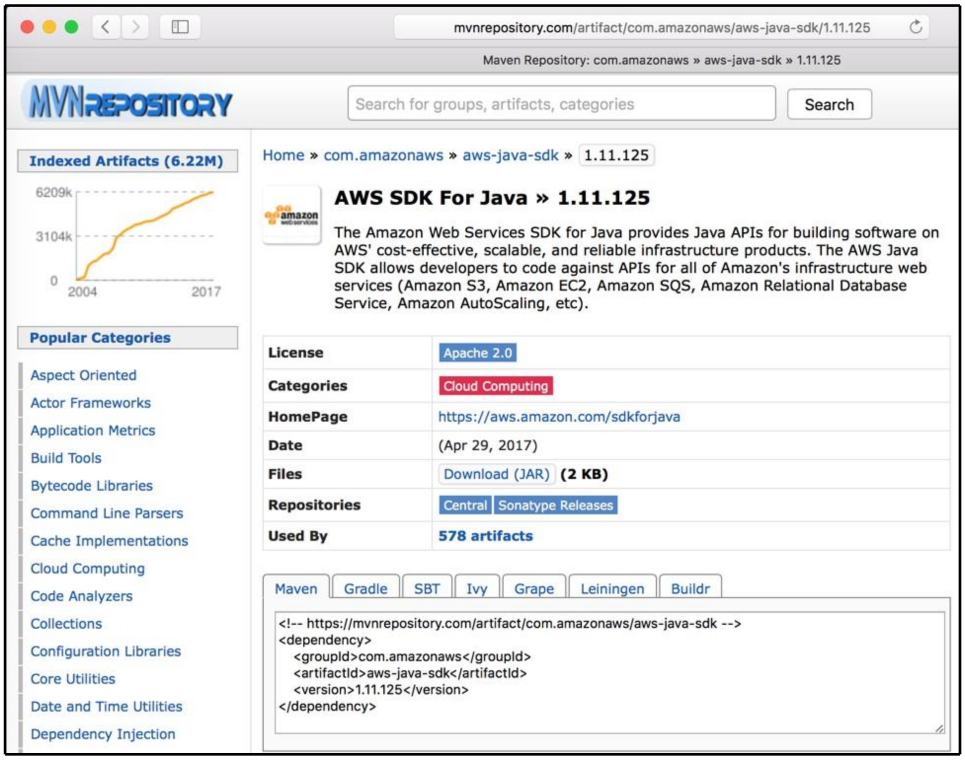
TIP

The `ldd` command is included in the `glibc-common` package on CentOS.

Package management

Java language and several lightweight programming languages such as Python, Ruby, and JavaScript have their own module or package management tool. For example, Maven (`http://maven.apache.org`) for Java, pip (`https://pip.pypa.io`) for Python, RubyGems (`https://rubygems.org`) for Ruby and npm (`https://www.npmjs.com`) for JavaScript.

Package management tool allows you to register your module or package to the centralized or private repository, and also allows to download the necessary packages. The following screenshot shows Maven repository for AWS SDK:



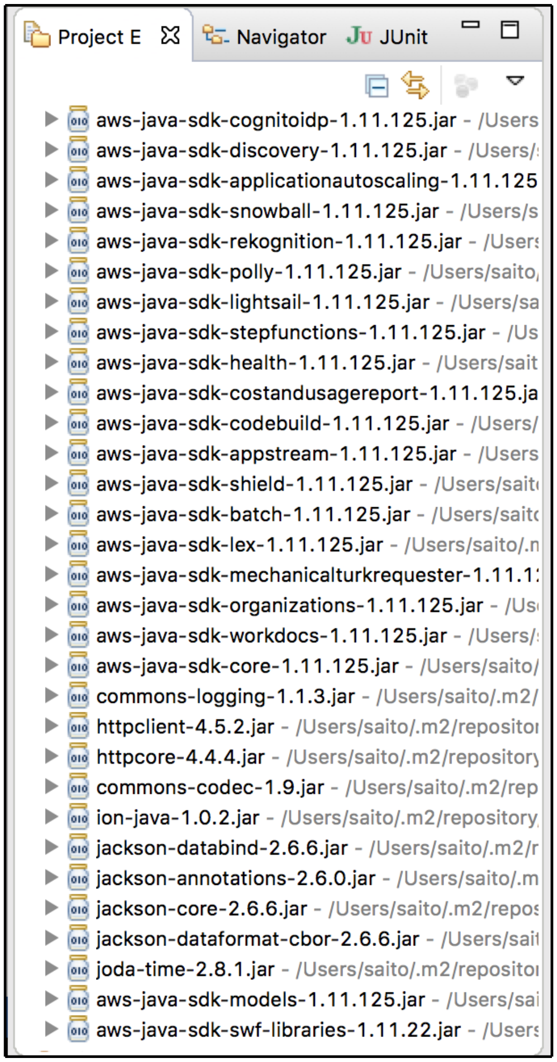
AWS SDK For Java » 1.11.125

The Amazon Web Services SDK for Java provides Java APIs for building software on AWS' cost-effective, scalable, and reliable infrastructure products. The AWS Java SDK allows developers to code against APIs for all of Amazon's infrastructure web services (Amazon S3, Amazon EC2, Amazon SQS, Amazon Relational Database Service, Amazon AutoScaling, etc).

License	Apache 2.0
Categories	Cloud Computing
HomePage	https://aws.amazon.com/sdkforjava
Date	(Apr 29, 2017)
Files	Download (JAR) (2 KB)
Repositories	Central Sonatype Releases
Used By	578 artifacts

```
<!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk -->
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk</artifactId>
    <version>1.11.125</version>
</dependency>
```

When you add some particular dependencies to your application, Maven downloads the necessary packages. The following screenshot is the result you get when you add `aws-java-sdk` dependency to your application:



```
aws-java-sdk-cognitoidp-1.11.125.jar - /Users
aws-java-sdk-discovery-1.11.125.jar - /Users/
aws-java-sdk-applicationautoscaling-1.11.125
aws-java-sdk-snowball-1.11.125.jar - /Users/s
aws-java-sdk-rekognition-1.11.125.jar - /Users
aws-java-sdk-polly-1.11.125.jar - /Users/saito,
aws-java-sdk-lightsail-1.11.125.jar - /Users/sa
aws-java-sdk-stepfunctions-1.11.125.jar - /Us
aws-java-sdk-health-1.11.125.jar - /Users/sait
aws-java-sdk-costandusagereport-1.11.125.ja
aws-java-sdk-codebuild-1.11.125.jar - /Users/
aws-java-sdk-appstream-1.11.125.jar - /Users
aws-java-sdk-shield-1.11.125.jar - /Users/sait
aws-java-sdk-batch-1.11.125.jar - /Users/saitc
aws-java-sdk-lex-1.11.125.jar - /Users/saito/.n
aws-java-sdk-mechanicalturkrequester-1.11.1
aws-java-sdk-organizations-1.11.125.jar - /Us
aws-java-sdk-workdocs-1.11.125.jar - /Users/
aws-java-sdk-core-1.11.125.jar - /Users/saito/
commons-logging-1.1.3.jar - /Users/saito/.m2/
httpclient-4.5.2.jar - /Users/saito/.m2/reposito
httpcore-4.4.4.jar - /Users/saito/.m2/repository
commons-codec-1.9.jar - /Users/saito/.m2/rep
ion-java-1.0.2.jar - /Users/saito/.m2/repository
jackson-databind-2.6.6.jar - /Users/saito/.m2/r
jackson-annotations-2.6.0.jar - /Users/saito/.m
jackson-core-2.6.6.jar - /Users/saito/.m2/repos
jackson-dataformat-cbor-2.6.6.jar - /Users/sai
joda-time-2.8.1.jar - /Users/saito/.m2/reposito
aws-java-sdk-models-1.11.125.jar - /Users/sai
aws-java-sdk-swf-libraries-1.11.22.jar - /Users
```

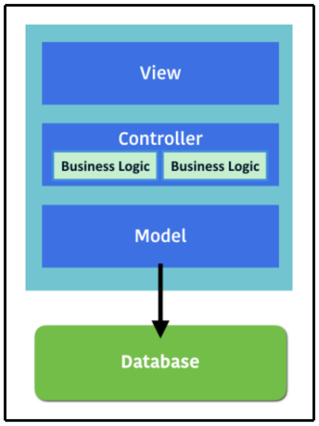
Modular programming helps you to increase software development speed and reduce it to reinvent the wheel, so it is the most popular way to develop a software application now.

However, applications need more and more combination of modules, packages, and frameworks, as and when we keep adding a feature and logic. This makes the application more complex and larger, especially server-side applications. This is because it usually needs to connect to a database such as RDBMS, as well as an authentication server such as LDAP, and then return the result to the user by HTML with the appropriate design.

Therefore, developers have adopted some software design patterns in order to develop an application with a bunch of modules within an application.

MVC design pattern

One of the popular application design patterns is **Model View and Controller** (**MVC**). It defines three layers. **View** layer is in charge of **user interface** (**UI**) **input output** (**I/O**). **Model** layer is in charge of data query and persistency such as load and store to database. Then, the **Controller** layer is in charge of business logic that is halfway between **View** and **Model**:



There are some frameworks that help developers to make MVC easier, such as Struts (`https://struts.apache.org/`), SpringMVC (`https://projects.spring.io/spring-framework/`), Ruby on Rails (`http://rubyonrails.org/`), and Django (`https://www.djangoproject.com/`). MVC is one of the successful software design pattern that is used for the foundation of modern web applications and services.

MVC defines a border line between every layer which allows many developers to jointly develop the same application. However, it causes side effects. That is, the size of the source code within the application keeps getting bigger. This is because database code (**Model**), presentation code (**View**), and business logic (**Controller**) are all within the same VCS repository. It eventually makes impact on the software development cycle, which gets slower again! It is called **monolithic**, which contains a lot of code that builds a giant exe/war program.

Monolithic application

There is no clear measurement of monolithic application definition, but it used to have more than 50 modules or packages, more than 50 database tables, and then it needs more than 30 minutes to build. When it needs to add or modify one module, it affects a lot of code, therefore developers try to minimize the application code change. This hesitation causes worse effects such that sometimes the application even dies because no one wants to maintain the code anymore.

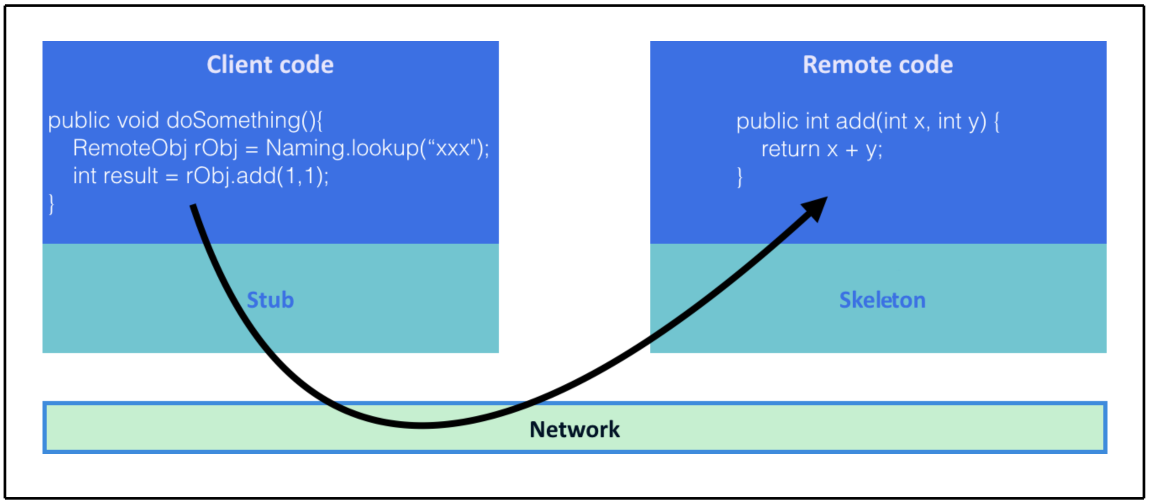
Therefore, the developer starts to divide monolithic applications in to small pieces of application and connect via the network.

Remote Procedure Call

Actually, dividing an application in to small pieces and connecting via the network has been attempted back in the 1990s. Sun Microsystems introduced **Sun RPC** (**Remote Procedure Call**). It allows you to use the module remotely. One of popular Sun RPC implementers is **Network File System** (**NFS**). CPU OS versions are independent across NFS client and NFS server, because they are based on Sun RPC.

The programming language itself also supports RPC-style functionality. UNIX and C language have the `rpcgen` tool. It helps the developer to generate a stub code, which is in charge of network communication code, so the developer can use the C function style and be relieved from difficult network layer programming.

Java has **Java Remote Method Invocation** (**RMI**) which is similar to Sun RPC, but for Java, **RMI compiler** (**rmic**) generates the stub code that connects remote Java processes to invoke the method and get a result back. The following diagram shows Java RMI procedure flow:



```
public void doSomething(){
    RemoteObj rObj = Naming.lookup("xxx");
    int result = rObj.add(1,1);
}
```

```
public int add(int x, int y) {
    return x + y;
}
```

Objective C also has **distributed object** and .NET has **remoting**, so most of the modern programming languages have the capability of Remote Procedure Call out of the box.

These Remote Procedure Call designs have the benefit to divide an application into multiple processes (programs). Individual programs can have separate source code repositories. It works well although machine resource (CPU, memory) was limited during 1990s and 2000s.

However, it was designed and intended to use the same programming language and also designed for client/server model architecture, instead of a distributed architecture. In addition, there was less security consideration; therefore, it is not recommended to use over a public network.

In the 2000s, there was an initiative **web services** that used **SOAP** (HTTP/SSL) as data transport, using XML as data presentation and service definition **Web Services Description Language** (**WSDL**), then used **Universal Description, Discovery, and Integration** (**UDDI**) as the service registry to look up a web services application. However, as the machine resources were not rich and due to the complexity of web services programming and maintainability, it is not widely accepted by developers.

RESTful design

Go to 2010s, now machine power and even the smartphone have plenty of CPU resource, in addition to network bandwidth of a few hundred Mbps everywhere. So, the developer starts to utilize these resources to make application code and system structure as easy as possible making the software development cycle quicker.

Based on hardware resources, it is a natural decision to use HTTP/SSL as RPC transport, but from having experience with web services difficulty, the developer makes it simple as follows:

- By making HTTP and SSL/TLS a standard transport
- By using HTTP method for **Create/Load/Upload/Delete** (**CLUD**) operation, such as `GET`/`POST`/`PUT`/`DELETE`
- By using URI as the resource identifier such as: user ID 123 as `/user/123/`
- By using JSON as the standard data presentation

It is called **RESTful** design, and that has been widely accepted by many developers and become de facto standard of distributed applications. RESTful application allows any programming language as it is HTTP-based, so the RESTful server is Java and client Python is very natural.

It brings freedom and opportunities to the developer that its easy to perform code refactoring, upgrade a library and even switch to another programming language. It also encourages the developer to build a distributed modular design by multiple RESTful applications, which is called microservices.

If you have multiple RESTful applications, there is a concern on how to manage multiple source code on VCS and how to deploy multiple RESTful servers. However, Continuous Integration, and Continuous Delivery automation makes a lower bar to build and deploy a multiple RESTful server application easier.

Therefore, microservices design is getting popular for web application developers.

Microservices

Although the name is micro, it is actually heavy enough compared to the applications from 1990s or 2000s. It uses full stack of HTTP/SSL server and contains entire MVC layers. The microservices design should care about the following topics:

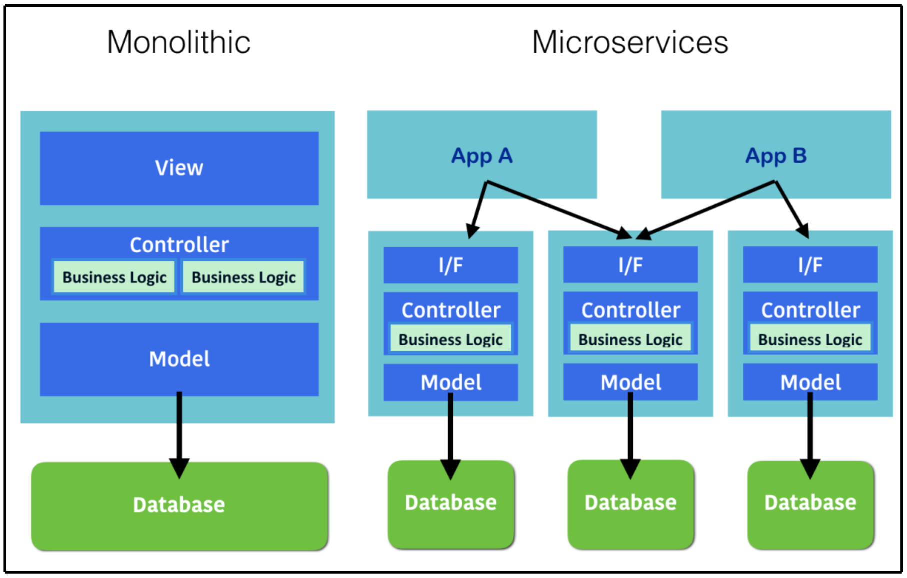
- **Stateless**: This doesn't store user session to the system, which helps to scale out easier.
- **No shared datastore**: The microservice should own the datastore such as database. It shouldn't share with the other application. It helps to encapsulate the backend database that is easy to refactor and update the database scheme within a single microservice.

- **Versioning and compatibility**: The microservice may change and update the API but should define a version and it should have backward compatibility. This helps to decouple between other microservices and applications.
- **Integrate CI/CD**: The microservice should adopt CI and CD process to eliminate management effort.

There are some frameworks that can help to build the microservice application such as Spring Boot (`https://projects.spring.io/spring-boot/`) and Flask (`http://flask.pocoo.org`). However, there are a lot of HTTP-based frameworks, so the developer can feel free to try and choose any preferred framework or even programming language. This is the beauty of the microservice design.

The following diagram is a comparison between monolithic application design and microservices design. It indicates that microservice (also MVC) design is the same as monolithic, which contains interface layer, business logic layer, model layer, and datastore.

But the difference is that the application (service) is constructed by multiple microservices and that different applications can share the same microservice underneath:



The developer can add the necessary microservice and modify an existing microservice with the rapid software delivery method that won't affect an existing application (service) anymore.

It is a breakthrough to an entire software development environment and methodology that is getting widely accepted by many developers now.

Although Continuous Integration and Continuous Delivery automation process helps to develop and deploy multiple microservices, the number of resources and complexity, such as Virtual Machine, OS, library, and disk volume and network can't compare with monolithic applications.

Therefore, there are some tools and roles that can support these large automation environments on the cloud.

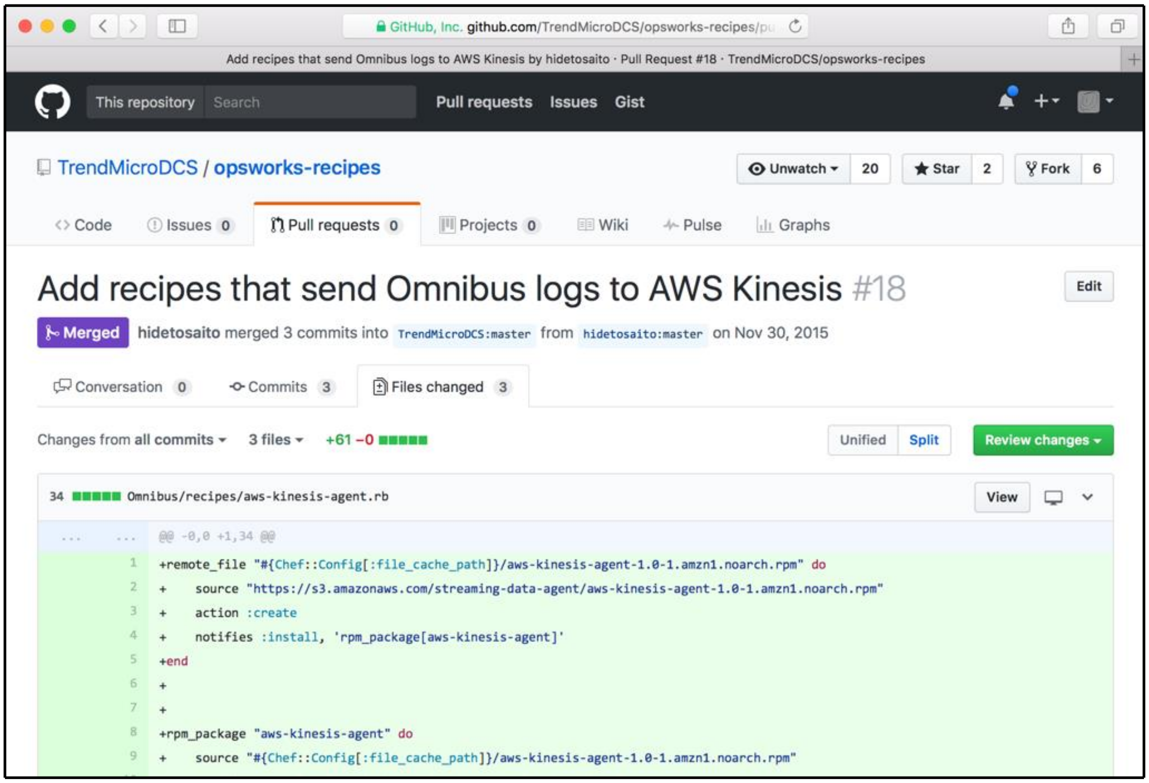
Automation and tools

As discussed previously, automation is the best practice to achieve rapid software delivery and solves the complexity to manage many microservices. However, automation tools are not an ordinary IT/infrastructure applications such as **Active Directory**, **BIND** (DNS), and **Sendmail** (MTA). In order to achieve automation, there is an engineer who should have both developer skill set to write a code, especially scripting language, and infrastructure operator skill set such as VM, network, and storage.

DevOps is a clipped compound of *development* and *operations* that can have an ability to make automation processes such as Continuous Integration, Infrastructure as code, and Continuous Delivery. DevOps uses some DevOps tools to make these automation processes.

Continuous Integration tool

One of the popular VCS tools is Git (`https://git-scm.com`). The developer uses Git to check-in and check-out the code all the time. There are some hosting Git service: GitHub (`https://github.com`) and Bitbucket (`https://bitbucket.org`). It allows you to create and save your Git repositories and collaborate with other users. The following screenshot is a sample pull request on GitHub:



Add recipes that send Omnibus logs to AWS Kinesis #18

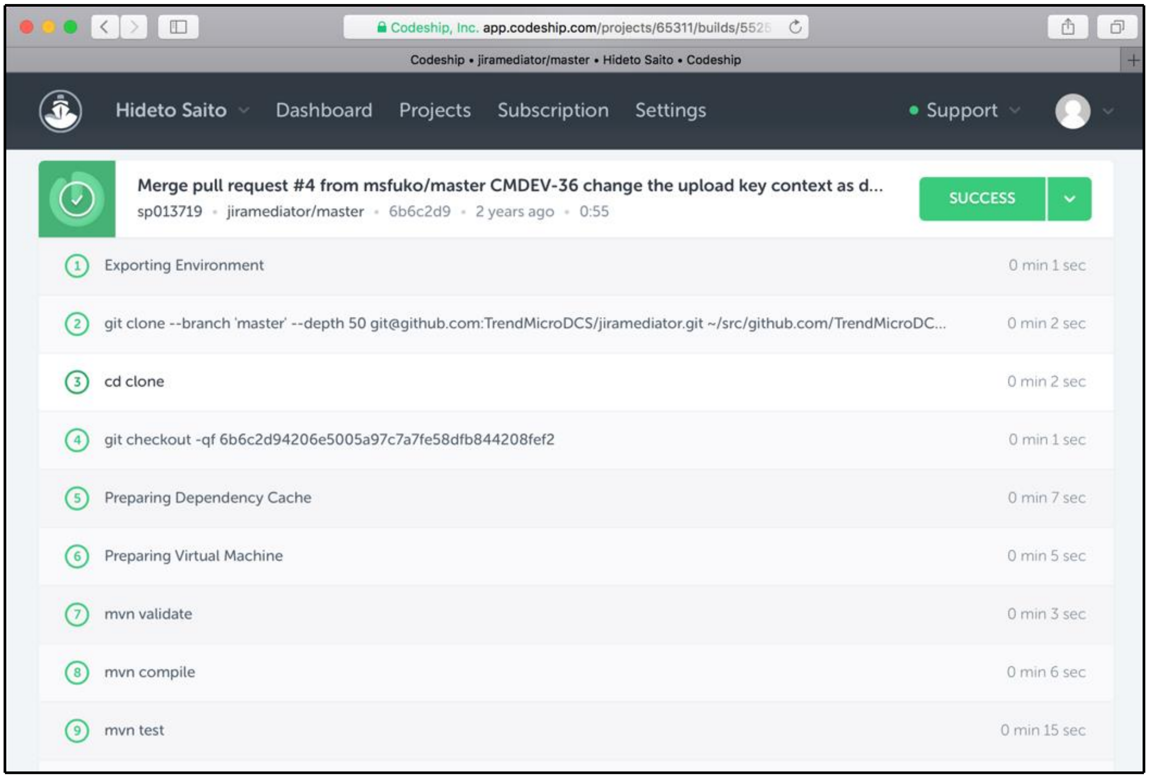
Merged hidetosaito merged 3 commits into `TrendMicroDCS:master` from `hidetosaito:master` on Nov 30, 2015

```
34 Omnibus/recipes/aws-kinesis-agent.rb
@@ -0,0 +1,34 @@
+remote_file "#{Chef::Config[:file_cache_path]}/aws-kinesis-agent-1.0-1.amzn1.noarch.rpm" do
+    source "https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-1.0-1.amzn1.noarch.rpm"
+    action :create
+    notifies :install, 'rpm_package[aws-kinesis-agent]'
+end
+
+
+rpm_package "aws-kinesis-agent" do
+    source "#{Chef::Config[:file_cache_path]}/aws-kinesis-agent-1.0-1.amzn1.noarch.rpm"
```

The build server has a lot of variation. Jenkins (`https://jenkins.io`) is one of well-established applications, which is the same as TeamCity (`https://www.jetbrains.com/teamcity/`). In addition to build server, you also have hosted services, the **Software as a Service (SaaS)** such as Codeship (`https://codeship.com`) and Travis CI (`https://travis-ci.org`). SaaS has the strength to integrate with other SaaS tools.

Build server is capable of invoking an external command such as a unit test program; therefore, build server is a key tool within CI pipeline.

The following screenshot is a sample build using Codeship; it checks out the code from GitHub and invokes Maven to build (`mvn compile`) and unit testing (`mvn test`):


Codeship, Inc. app.codeship.com/projects/65311/builds/552
Codeship • jiramediator/master • Hideto Saito • Codeship
Hideto Saito
Dashboard
Projects
Subscription
Settings
Support
SUCCESS

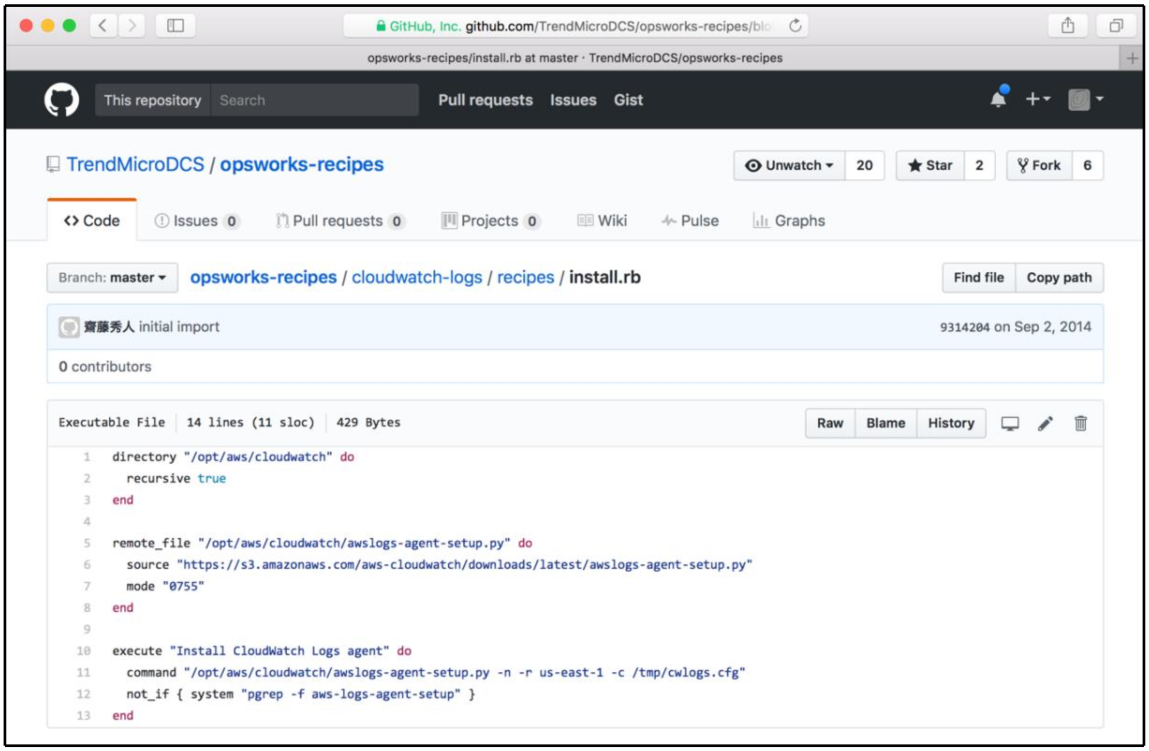
Merge pull request #4 from msfuko/master CMDEV-36 change the upload key context as d...
sp013719 • jiramediator/master • 6b6c2d9 • 2 years ago • 0:55

```
1 Exporting Environment                                                                 0 min 1 sec
2 git clone --branch 'master' --depth 50 git@github.com:TrendMicroDCS/jiramediator.git ~/src/github.com/TrendMicroDC...   0 min 2 sec
3 cd clone                                                                              0 min 2 sec
4 git checkout -qf 6b6c2d94206e5005a97c7a7fe58dfb844208fef2                             0 min 1 sec
5 Preparing Dependency Cache                                                            0 min 7 sec
6 Preparing Virtual Machine                                                             0 min 5 sec
7 mvn validate                                                                          0 min 3 sec
8 mvn compile                                                                           0 min 6 sec
9 mvn test                                                                              0 min 15 sec
```

Continuous Delivery tool

There are a variety of configuration management tools such as Puppet (`https://puppet.com`), Chef (`https://www.chef.io`), and Ansible (`https://www.ansible.com`), which are the most popular in configuration management.

AWS OpsWorks (`https://aws.amazon.com/opsworks/`) provides a managed Chef platform. The following screenshot is a Chef recipe (configuration) of installation of Amazon CloudWatch Log agent using AWS OpsWorks. It automates to install CloudWatch Log agent when launching an EC2 instance:

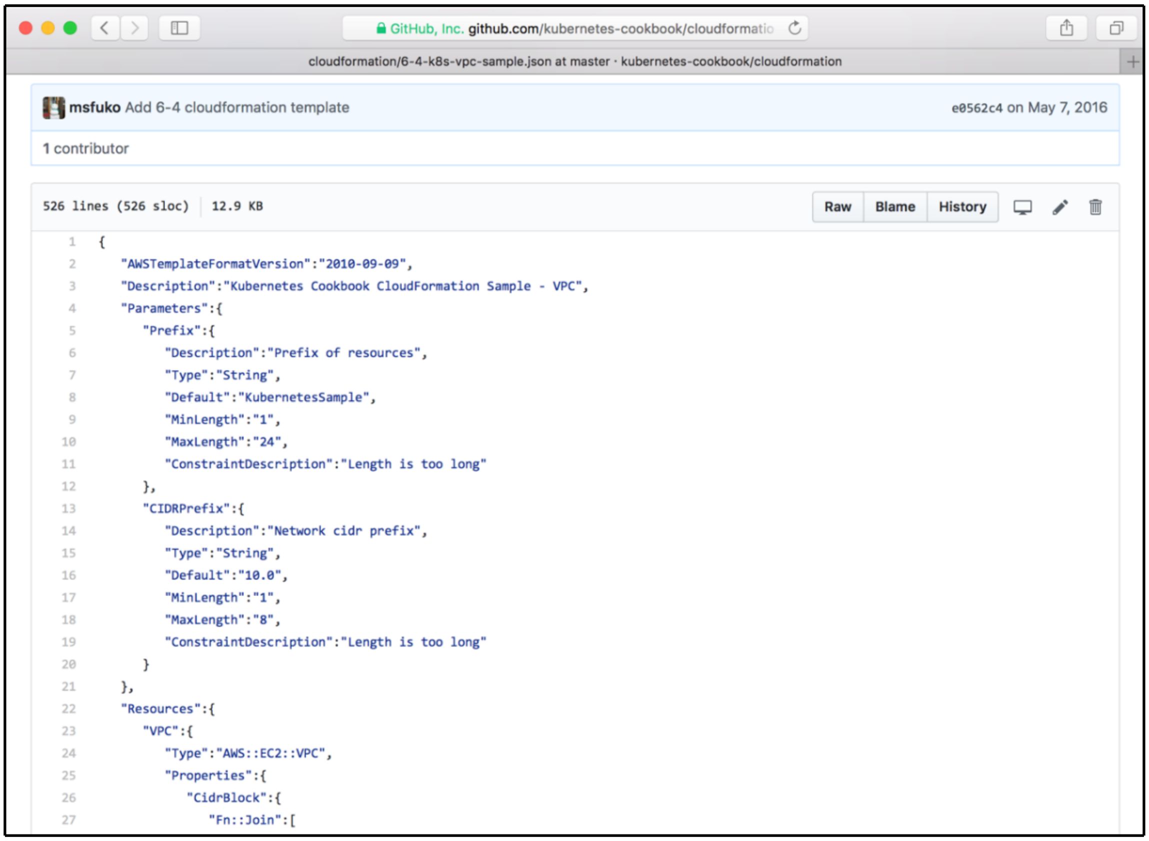

GitHub, Inc. github.com/TrendMicroDCS/opsworks-recipes/blo
opsworks-recipes/install.rb at master · TrendMicroDCS/opsworks-recipes
This repository
Search
Pull requests
Issues
Gist
TrendMicroDCS / opsworks-recipes
Unwatch 20
Star 2
Fork 6
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Pulse
Graphs
Branch: master
opsworks-recipes / cloudwatch-logs / recipes / install.rb
Find file
Copy path
齋藤秀人 initial import
9314204 on Sep 2, 2014
0 contributors
Executable File 14 lines (11 sloc) 429 Bytes
Raw
Blame
History

```
directory "/opt/aws/cloudwatch" do
  recursive true
end

remote_file "/opt/aws/cloudwatch/awslogs-agent-setup.py" do
  source "https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py"
  mode "0755"
end

execute "Install CloudWatch Logs agent" do
  command "/opt/aws/cloudwatch/awslogs-agent-setup.py -n -r us-east-1 -c /tmp/cwlogs.cfg"
  not_if { system "pgrep -f aws-logs-agent-setup" }
end
```

AWS CloudFormation (`https://aws.amazon.com/cloudformation/`) helps to achieve infrastructure as code. It supports the automation for AWS operation, for example, to perform the following functions:

1. Creating a VPC.
2. Creating a subnet on VPC.
3. Creating an internet gateway on VPC.
4. Creating a routing table to associate a subnet to the internet gateway.
5. Creating a security group.
6. Creating a VM instance.
7. Associating a security group to a VM instance.

The configuration of CloudFormation is written by JSON as shown in the following screenshot:


GitHub, Inc. github.com/kubernetes-cookbook/cloudformatio
cloudformation/6-4-k8s-vpc-sample.json at master · kubernetes-cookbook/cloudformation
msfuko Add 6-4 cloudformation template
e0562c4 on May 7, 2016
1 contributor
526 lines (526 sloc) 12.9 KB
Raw
Blame
History

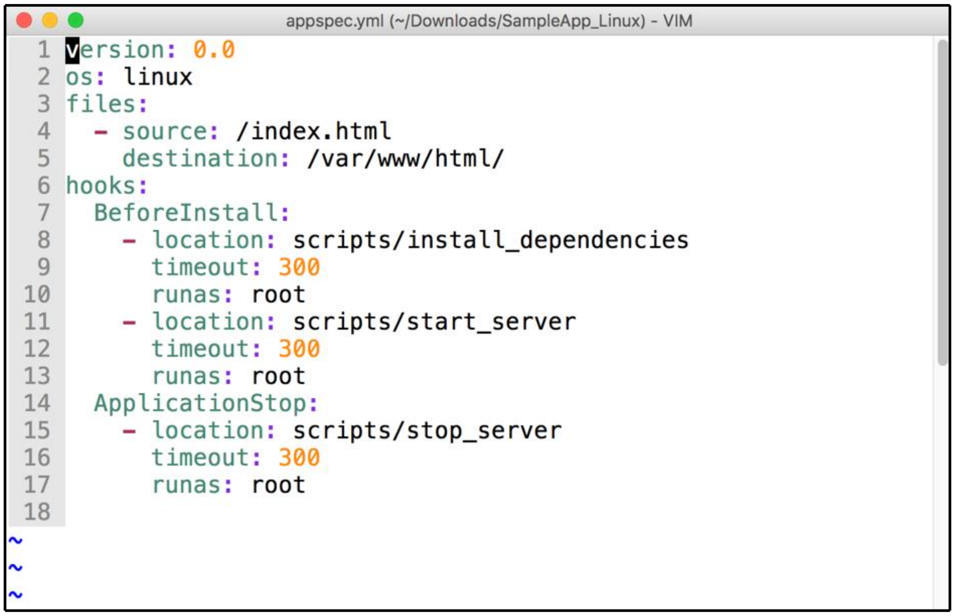
```
{
  "AWSTemplateFormatVersion":"2010-09-09",
  "Description":"Kubernetes Cookbook CloudFormation Sample - VPC",
  "Parameters":{
    "Prefix":{
      "Description":"Prefix of resources",
      "Type":"String",
      "Default":"KubernetesSample",
      "MinLength":"1",
      "MaxLength":"24",
      "ConstraintDescription":"Length is too long"
    },
    "CIDRPrefix":{
      "Description":"Network cidr prefix",
      "Type":"String",
      "Default":"10.0",
      "MinLength":"1",
      "MaxLength":"8",
      "ConstraintDescription":"Length is too long"
    }
  },
  "Resources":{
    "VPC":{
      "Type":"AWS::EC2::VPC",
      "Properties":{
        "CidrBlock":{
          "Fn::Join":[
```

It supports parameterize, so it is easy to create an additional environment with different parameters (for example, VPC and CIDR) using a JSON file with the same configuration. In addition, it supports the update operation. So, if there is a need to change a part of the infrastructure, there's no need to recreate. CloudFormation can identify a delta of configuration and perform only the necessary infrastructure operations on behalf of you.

AWS CodeDeploy (`https://aws.amazon.com/codedeploy/`) is also a useful automation tool. But focus on software deployment. It allows the user to define. The following are some actions onto the YAML file:

1. Where to download and install.
2. How to stop the application.
3. How to install the application.
4. After installation, how to start and configure an application.

The following screenshot is an example of AWS CodeDeploy configuration file `appspec.yml`:

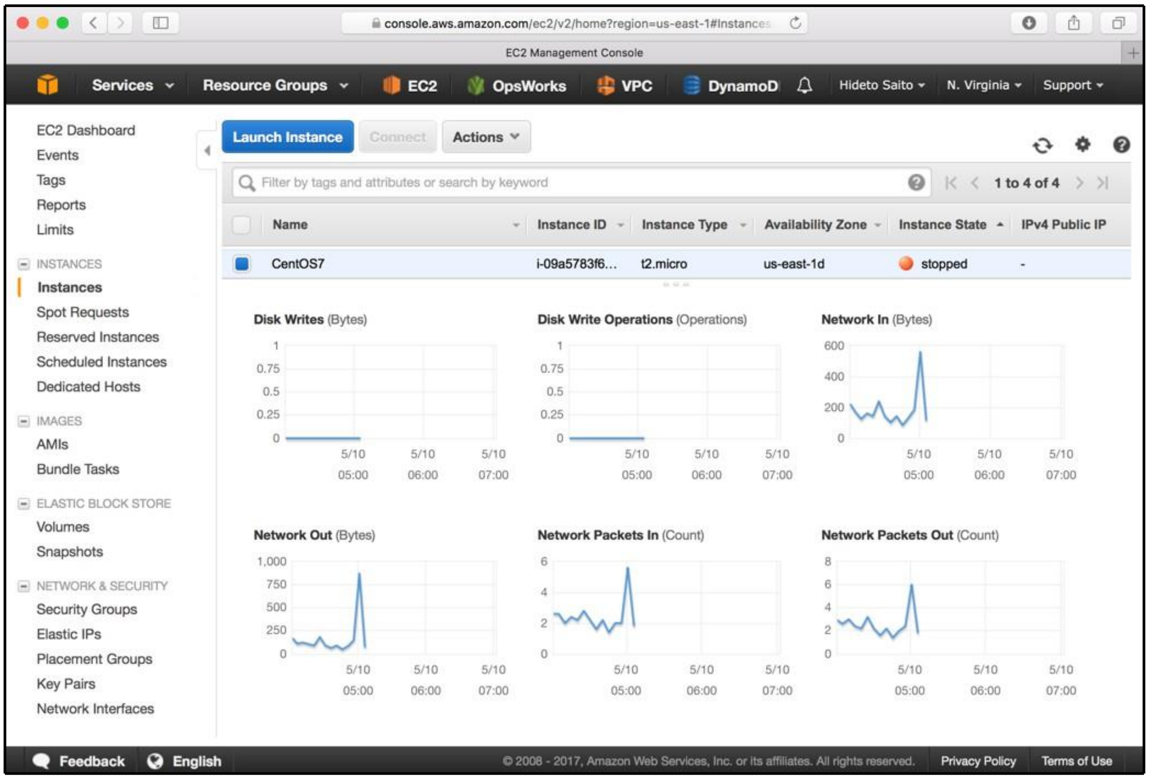


```yaml
version: 0.0
os: linux
files:
  - source: /index.html
    destination: /var/www/html/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies
      timeout: 300
      runas: root
    - location: scripts/start_server
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server
      timeout: 300
      runas: root

~
~
~
```

Monitoring and logging tool

Once you start to manage some microservices using a cloud infrastructure, there are some monitoring tools that help you to manage your servers.

Amazon CloudWatch is the built-in monitoring tool on AWS. No agent installation is needed; it automatically gathers some metrics from AWS instances and visualizes for DevOps. It also supports to set an alert based on the criteria that you set. The following screenshot is an Amazon CloudWatch metrics for EC2 instance:

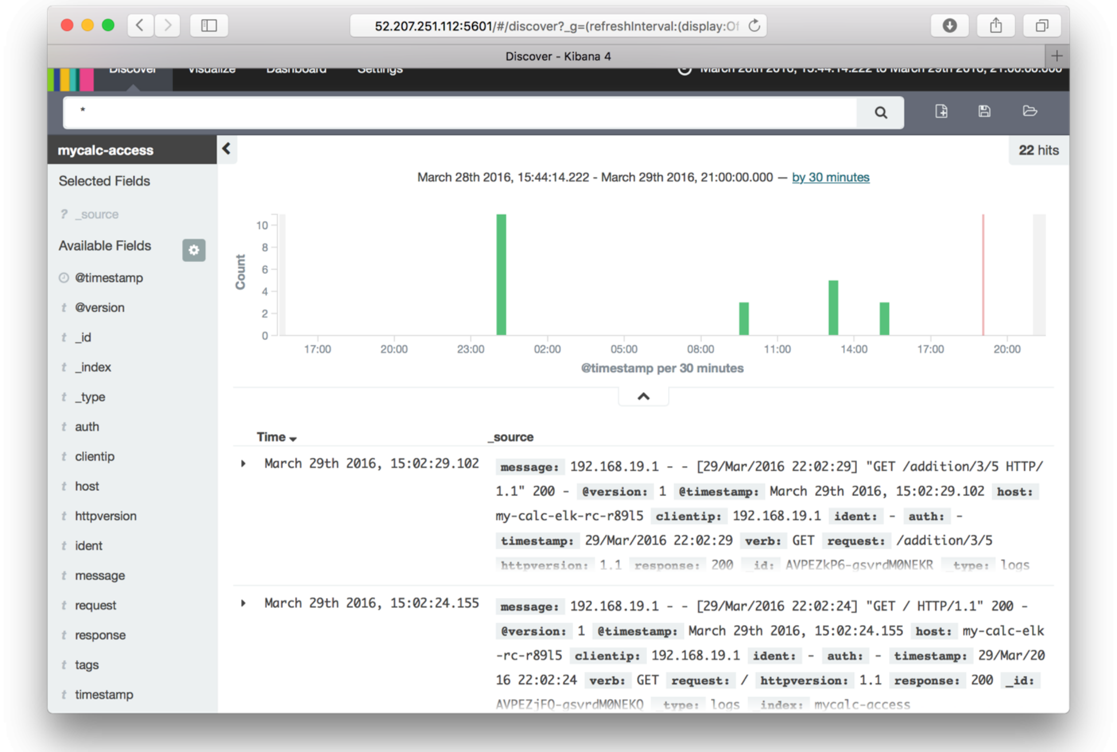


Name	Instance ID	Instance Type	Availability Zone	Instance State	IPv4 Public IP
CentOS7	i-09a5783f6...	t2.micro	us-east-1d	stopped	-

Amazon CloudWatch also supports to gather an application log. It requires installing an agent on EC2 instance; however, centralized log management is useful when you need to start managing multiple microservice instances.

ELK is a popular combination of stack that stands for Elasticsearch (`https://www.elastic.co/products/elasticsearch`), Logstash (`https://www.elastic.co/products/logstash`), and Kibana (`https://www.elastic.co/products/kibana`). Logstash helps to aggregate the application log and transform to JSON format and then send to Elasticsearch.

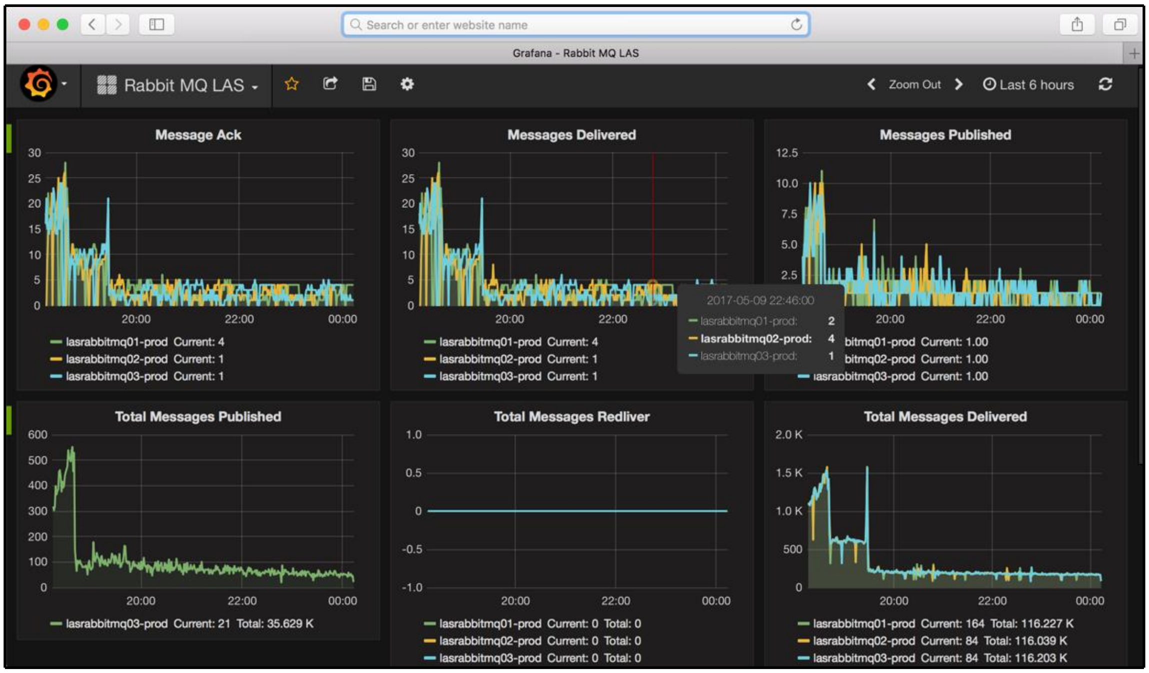
Elasticsearch is a distributed JSON database. Kibana can visualize the data, which is stored on Elasticsearch. The following example is a Kibana, which shows Nginx access log:



```
Time    _source
March 29th 2016, 15:02:29.102  message: 192.168.19.1 - - [29/Mar/2016 22:02:29] "GET /addition/3/5 HTTP/
                               1.1" 200 - @version: 1 @timestamp: March 29th 2016, 15:02:29.102 host:
                               my-calc-elk-rc-r8915 clientip: 192.168.19.1 ident: - auth: -
                               timestamp: 29/Mar/2016 22:02:29 verb: GET request: /addition/3/5
                               httpversion: 1.1 response: 200 _id: AVPEZkP6-gsvrdM0NEKR _type: logs
March 29th 2016, 15:02:24.155  message: 192.168.19.1 - - [29/Mar/2016 22:02:24] "GET / HTTP/1.1" 200 -
                               @version: 1 @timestamp: March 29th 2016, 15:02:24.155 host: my-calc-elk
                               -rc-r8915 clientip: 192.168.19.1 ident: - auth: - timestamp: 29/Mar/20
                               16 22:02:24 verb: GET request: / httpversion: 1.1 response: 200 _id:
                               AVPEZjFQ-gsvrdM0NEKQ _type: logs _index: mycalc-access
```

Grafana (`https://grafana.com`) is another popular visualization tool. It used to be connected with time series database such as Graphite (`https://graphiteapp.org`) or InfluxDB (`https://www.influxdata.com`). Time series database is designed to store the data, which is flat and de-normalized numeric data such as CPU usage and network traffic. Unlike RDBMS, time series database has some optimization to save the data space and faster query for numeric data history. Most of DevOps monitoring tools are using time series database in the backend.

The following example is a Grafana that shows **Message Queue Server** statistics:


Search or enter website name
Grafana - Rabbit MQ LAS
Rabbit MQ LAS
Zoom Out
Last 6 hours
Message Ack
lasrabbitmq01-prod Current: 4
lasrabbitmq02-prod Current: 1
lasrabbitmq03-prod Current: 1
Messages Delivered
lasrabbitmq01-prod Current: 4
lasrabbitmq02-prod Current: 1
lasrabbitmq03-prod Current: 1
2017-05-09 22:46:00
lasrabbitmq01-prod: 2
lasrabbitmq02-prod: 4
lasrabbitmq03-prod: 1
Messages Published
bitmq01-prod Current: 1.00
bitmq02-prod Current: 1.00
lasrabbitmq03-prod Current: 1.00
Total Messages Published
lasrabbitmq03-prod Current: 21 Total: 35.629 K
Total Messages Redliver
lasrabbitmq01-prod Current: 0 Total: 0
lasrabbitmq02-prod Current: 0 Total: 0
lasrabbitmq03-prod Current: 0 Total: 0
Total Messages Delivered
lasrabbitmq01-prod Current: 164 Total: 116.227 K
lasrabbitmq02-prod Current: 84 Total: 116.039 K
lasrabbitmq03-prod Current: 84 Total: 116.203 K

Communication tool

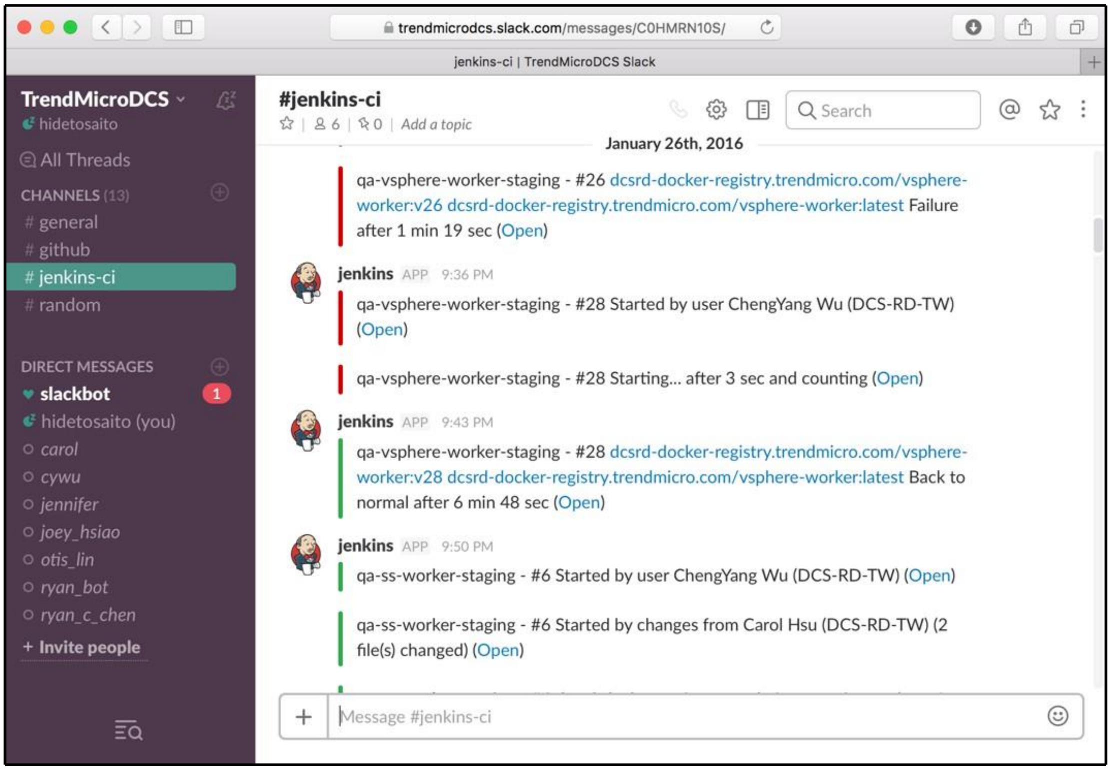
Once you start to use several DevOps tools as we saw earlier, you need to go back and forth to visit several consoles to check whether CI and CD pipelines work properly or not. For example, consider the following points:

1. Merge the source code to GitHub.
2. Trigger the new build on Jenkins.
3. Trigger AWS CodeDeploy to deploy the new version of the application.

These events need to be tracked by time sequence, and if there are some troubles, DevOps needs to discuss it with the developer and QA to handle the cases. However, there are some over-communication needs, because DevOps needs to capture the event one by one and then explain, probably via e-mail. It is not efficient and in the meantime the issue is still going on.

There are some communication tools that help to integrate these DevOps tools and anyone can join to look at the event and comment to each other. Slack (`https://slack.com`) and HipChat (`https://www.hipchat.com`) are the most popular communication tools.

These tools support to integrate to SaaS services so that DevOps can see the event on the single chat room. The following screenshot is a Slack chat room that integrates with Jenkins:



January 26th, 2016

qa-vsphere-worker-staging - #26 dcsrd-docker-registry.trendmicro.com/vsphere-worker:v26 dcsrd-docker-registry.trendmicro.com/vsphere-worker:latest Failure after 1 min 19 sec (Open)

jenkins APP 9:36 PM
qa-vsphere-worker-staging - #28 Started by user ChengYang Wu (DCS-RD-TW) (Open)

qa-vsphere-worker-staging - #28 Starting... after 3 sec and counting (Open)

jenkins APP 9:43 PM
qa-vsphere-worker-staging - #28 dcsrd-docker-registry.trendmicro.com/vsphere-worker:v28 dcsrd-docker-registry.trendmicro.com/vsphere-worker:latest Back to normal after 6 min 48 sec (Open)

jenkins APP 9:50 PM
qa-ss-worker-staging - #6 Started by user ChengYang Wu (DCS-RD-TW) (Open)

qa-ss-worker-staging - #6 Started by changes from Carol Hsu (DCS-RD-TW) (2 file(s) changed) (Open)

Public cloud

CI CD and automation work can be achieved easily when used with cloud technology. Especially public cloud API helps DevOps to come up with many CI CD tools. Public cloud such as Amazon Web Services (`https://aws.amazon.com`) and Google Cloud Platform (`https://cloud.google.com`) provides some APIs to DevOps to control the cloud infrastructure. DevOps can be a relief from capacity and resource limitation, just pay as you go whenever the resource is needed.

Note:

- This is a **preview DevOps Kubernetes e-Book** containing **only 30 pages**.
- It is provided to help you understand **how the full DevOps Kubernetes e-Book and is structured**.
- The **complete DevOps Kubernetes e-Book** includes detailed concepts, real-world examples, and career guidance.
- **Purchase the full DevOps Kubernetes e-Book for just ₹349**.
- Buy now from our official website: https://topitcourses.com/devops-kubernetes-ebook/