IT Automation using Ansible

***Abstract*— The increasing number of cars in cities can cause high volume of traffic, and implies that traffic violations become more critical nowadays around the world. This causes severe destruction of property and more accidents that may endanger the lives of the people. To solve the alarming problem and prevent such unfathomable consequences, traffic violation detection systems are needed. For which the system enforces proper traffic regulations at all times, and apprehend those who does not comply.**

**A traffic violation detection system must be realized in real-time as the authorities track the roads all the time. Hence, traffic enforcers will not only be at ease in implementing safe roads accurately, but also efficiently; as the traffic detection system detects violations faster than humans. The goal of the project is to automate the traffic signal violation detection system and make it easy for the traffic police department to monitor the traffic and take action against the violated vehicle owner in a fast and efficient way. Detecting and tracking the vehicle and their activities accurately is the main priority of the system**

**INTRODUCTION**

The word “DevOps” was coined in 2009 by Patrick Debois. The term was formed by combining “development” and “operations,” which provides a starting point for understanding exactly what people typically mean when they say “DevOps”. It is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. From an academic perspective, Len Bass, Ingo Weber, and Liming Zhu—three computer science researchers from the **Commonwealth Scientific and Industrial Research Organisation (CSIRO)** and the Software Engineering Institute—suggested defining DevOps as "a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality". DevOps implementations utilize technology— especially automation tools that can leverage an increasingly programmable and dynamic infrastructure from a life cycle perspective.”

The two primary antecedents of DevOps are:

**Enterprise systems management (ESM).** Many of the people involved in the initial definition of DevOps were system administrators. These operations experts brought key ESM best practices to DevOps, including configuration management, system monitoring, automated provisioning, and the tool chain approach.

**Agile development**. “DevOps can be interpreted as an outgrowth of Agile—agile software development prescribes close collaboration of customers, product management.

The DevOps aims to improve communication, Collaboration, and integration between software developers (Dev) and IT operations professionals (Ops). As the part of the DevOps, certain actions take over standard tools from software development area e.g., code version systems or

code-revision management to manage what is these days known as Infrastructure-as-Code(IaC).DevOps integration targets product delivery, continuous testing, quality testing, feature development, and maintenance releases in order to improve reliability and security and provide faster development and deployment cycles. Many of the ideas involved in DevOps came from the enterprise systems management and agile software development movements. The goals of DevOps span the entire delivery pipeline. They include: Improved deployment frequency, Faster market, Lower failure rate of new releases, Shortened lead time between fixes; Faster mean time to recovery (in the event of a new release crashing or otherwise disabling the current system).

**IT automation** is the process of creating software and systems to replace repeatable processes and reduce manual intervention. It accelerates the delivery of IT infrastructure and applications by automating manual processes that previously required a human touch. With IT automation, software is used to set up and repeat instructions, processes, or policies that save time and free up IT staff for more strategic work. With the rise of virtualized networks and cloud services that require rapid, complex provisioning, automation is an indispensable strategy for helping IT teams deliver services with improved speed, consistency, and security. IT automation is a powerful tool that can scale a business, provide significant cost savings, and allow IT staff to focus on strategic rather than administrative work. A wide range of data center and cloud operations can be automated, resulting in faster operations. Thanks to automation, IT environments can scale more quickly with fewer errors and are more responsive to business needs. A fully automated environment can reduce the time to delivery for production-ready resources from weeks to less than a day.IT automation software can perform a range of IT tasks and processes, from simple to complex.

The word “**Ansible**” is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs. Designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time. It uses no agents and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (YAML).

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates. Ansible’s main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport, and a language that is designed around auditability by humans–even those not familiar with the program. It's a simple automation language that can perfectly describe an IT application infrastructure in Ansible Playbooks. It's an automation engine that runs Ansible Playbooks.

Ansible Tower is an enterprise framework for controlling, securing and managing your Ansible automation with a UI and Restful API.

Ansible is

1. Simple: Human readable automation. No special coding skills needed. Tasks executes in order.

2.Powerful: Application development, configuration management workflow orchestration.

3.Agentless : Agentless architecture, no agent to exploit or update, uses OpenSSH, more efficient and more secure.

**PROBLEM STATEMENT**

Installation, updating are some of the common and daily routines that are carried out with systems and in IT world. Also installing new software or updating the exiting one with

new version is considerable with single system. When it comes to point of installation and updating the entire systems under a server or whole systems of company is time consuming and also an tedious work. So here comes the idea and utilization of Ansible. Ansible is an IT automation tool.You won’t need to write custom code to automate your systems; you list the tasks required to be done by writing a playbook, and Ansible will figure out how to get your systems to the state you want them to be in. Playbooks are the files where Ansible code is written. Playbooks are written in YAML format.

YAML stands for Yet another Mark-up Language. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks. Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

**OBJECTIVE**

Automation with Ansible is designed for Linux system administrators and developers who need to automate provisioning, configuration, application deployment, and orchestration. Incorporating IT automation is key to managing large numbers of systems and applications efficiently and consistently at scale. In this proposed system we will write Ansible playbooks to automate tasks, and will run them to ensure servers are correctly deployed and configured.

1. To build system to automate the infrastructure using Ansible.

2. To implement the auto backup solution using Cron job.

**PURPOSE OF ANSIBLE**

Ansible is designed to be very simple, reliable, and consistent for configuration management. Ansible configurations are simple data descriptions of infrastructure and are both readable by humans and parsable by machines. All you need to start managing systems is a password or an SSH (Secure Socket Shell, a network protocol) key. Ansible lets you quickly and easily deploy multitier apps. You won’t need to write custom code to automate your systems; you list the tasks required to be done by writing a playbook, and Ansible will figure out how to

get your systems to the state you want them to be in. In other words, you won’t have to configure the applications on every machine manually. When you run a playbook from your control machine, Ansible uses SSH to communicate with the remote hosts and run all the commands (tasks).

**LITERATURE SURVEY**

**Cloud Computing:** Cloud computing is a paradigm of distributed computing to provide the customers on-demand, utility based computing services. Cloud users can provide more reliable, availableand updated services to their clients in turn. Cloud itself consists of physical machines in thedata centers of cloud providers. Virtualization is provided on top of these physical machines.These virtual machines are provided to the cloud users. Different cloud provider providescloud services of different abstraction level. E.g. Amazon EC2 enables the users to handle

very low level details where Google App-Engine provides a development platform for the developers to develop their applications. So the cloud services are divided into many types

like Software as a Service, Platform as a Service or Infrastructure as a Service. These services are available over the Internet in the whole world where the cloud acts as the single point of access for serving all customers. Cloud computing architecture addresses difficulties of large scale data processing.

Types of Cloud : It can be of three types.

1. Private Cloud – This type of cloud is maintained within an organization and used solely for their internal purpose. So the utility model is not a big term in this scenario. Many companies are moving towards this setting and experts consider this is the 1st step for an organization to move into cloud. Security, network bandwidth are not critical issues for private cloud.

2. Public Cloud – In this type an organization rents cloud services from cloud providers on-demand basis. Services provided to the users using utility computing model.

3. Hybrid Cloud – This type of cloud is composed of multiple internal or external cloud. This is the scenario when an organization moves to public cloud computing domain from its internal private cloud**.**

Cloud –Types Public cloud:

Public cloud or external cloud describes cloud computing in the traditional mainstream. Public clouds are run by third parties, and applications from different customers are likely to be mixed together on the cloud’s servers, storage systems, and networks.

A public cloud provides services to multiple customers. Hybrid cloud: Hybrid clouds combine both public and private cloud models. This is most often seen with the use of storage clouds to support Web 2.0 applications. Private cloud: Private clouds are built for the exclusive use of one client, providing the utmost control over data, security, and quality of service. The company owns the infrastructure and has control over how applications are deployed on it. Private clouds can be built and managed by a company’s own IT organization or by a cloud provider . Cloud computing products and services.

They can be classified into 4 major categories:

1. Application as service ( AaaS)

2. Platform as a Service (PaaS)

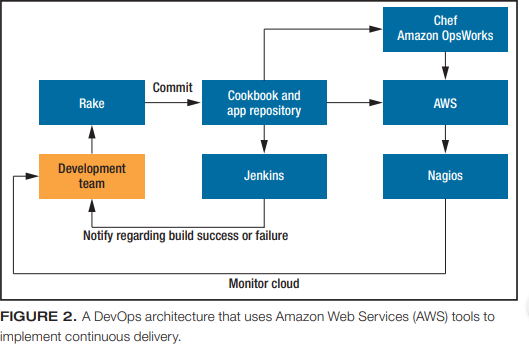
3. Infrastructure as a service (IaaS)

4. Software as a Service (SaaS)

1. Application as s service (AaaS): These are the first kind of cloud computing services that came into being. Under this, a service is made available to an end-user. The enduser is asked to create an account with the service provider and start using the application. One of first famous application was web-based email service by hotmail started in 1996. Scores of such services are available now on the web.

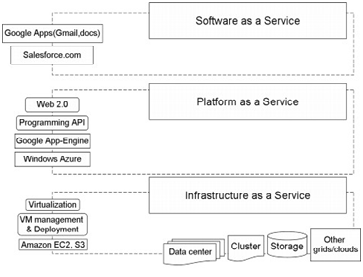
2. Platform as a Service (PaaS): Cloud vendors are companies that offer cloud computing services and products. One of the services that they provide is called PaaS. Under this a computing platform such as operating system is provided to a customer or end user on a monthly rental basis. Some of the major cloud computing vendor are Amazon, Microsoft, Google etc.

3. Infrastructure as a service: The cloud computing vendors offer infrastructure as a service. One may avail hardware services such as processors, memory, networks etc on agreed basis for specific duration and price.

4. Software as a service (SaaS): Software package such as CRM or CAD/CAM can be accessed under cloud computing scheme. Here a customer upon registration is allowed to use software accessible through net and use it for his or his business process. The related data and work may be stored on local machines or with the service providers. SaaS services may be available on rental basis or on per use basis.

Working Of Cloud Computing: Cloud Computing system can be divided it into two sections: the front end and the back end. They connect to each other through a network, usually the Internet. The front end is the side the computer user, or client, sees.The back end is the "cloud" section of the system. On the back end there are various computers,servers and data storage systems that create the "cloud" of computing services.A central server administers the system, monitoring traffic and client demands to ensure everything runs smoothly. It followsa set of rules called protocols Servers and remote computers do most of

the work and store the data.



**What is DevOps?**

With modern businesses moving at the speed of cloud, DevOps has become an increasingly common approach to software delivery that development and operations teams use to build, test, deploy, and monitor applications with speed, quality, and control.

DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing marketplace demands. It is an approach on the journey to lean and agile software delivery that promotes closer collaboration between lines of business ,development, and IT operations while removing barriers between your stakeholders, and your customers.

To be essential to customers, all stakeholders in the delivery process need to collaborate. Development teams need to design, develop, deliver and run the software as quickly and reliably as possible. Operations teams need to identify and resolve problems as soon as possible by monitoring, predicting failure, managing the environment and fixing issues. Combining this common approach across Dev and Ops with the ability to monitor and analyze bottlenecks and optimize as quickly as possible gives you DevOps—a collaborative approach across business, development, and operation stakeholders to deliver and run reliable software as

soon as possible.

**DevOps tools**

DevOps tools cover a range of processes within the software development life cycle:

• Define and plan, which focuses on planning DevOps workflows for iterations, release management, and issue tracking. Notable tools or tool vendors in this space include Atlassian, CA Technologies, IBM, iRise, and Jama Software.

• Code, build, and configure, which focuses on code development and review, source code management, and code merging. Notable tools/tool vendors include BitBucket, Electric Cloud, GitLab, GitHub, and IBM.

• Test, which verifies that the quality of the software release and code are maintained throughout the development process and that the highest quality deploys to production. Notable tools/tool vendors include Delphix, FlawCheck, HP, IBM, Microsoft, Parasoft, SonarSource, Skytap, and ThoughtWorks.

• Packaging and preproduction, which refers to the activities involved once the release is ready for deployment; it’s also called staging or preproduction. Notable tools/tool vendors include IBM, Inedo’s ProGet, Jfrog’s Artifactory, Sonatype Nexus repository.

Release, deploy, and orchestration, which is the process of actually releasing software and usually involves change management, release approvals, release automation, schedule orchestration, provisioning, and deploying into production.

•Continuous management and configuration includes continuous configuration automation, configuration management, and infrastructure as code. Notable tools/tool vendors include Ansible, Chef, IBM, Puppet Labs, Otter, and Salt.

•Monitoring reports application performance and helps identify issues impacting the user experience. Tools/tool vendors include Big Panda, IBM, New Relic, Plumbr, and Wireshark.

**DEVOPS METHODOLOGIES**

DevOps grew out of Agile. Agile is a way of producing software in short iterations on a continuous delivery schedule of new features and bug fixes in rapid cycles from two to four weeks. In contrast, DevOps brings the development and operations teams together to focus on eliminating silos to decrease time of addressing customer feedback and break down bottlenecks to enable continuous software delivery. Consequently, they can build, test, and release software more quickly with as much efficiency and speed as possible. Not only does DevOps involve more of the organization in the development process—including lines of business, suppliers involved in software delivery, and consumers themselves—but it does it in a way that speeds development and improves quality, according to a white paper on software-driven innovation. This can lead to the creation of a culture of innovation when you adopt DevOps methodologies, allowing you to collaborate and react with agility to changes in the market.

DevOps methodologies include the following:

- [Continuous integration](https://www.ibm.com/cloud/learn/continuous-integration), which is where coding, building, integrating, and testing take place.

- [Continuous delivery,](https://www.ibm.com/cloud/learn/continuous-delivery) which includes continuous integration, but mainly focuses on product releases.

- [Continuous deployment](https://www.ibm.com/cloud/learn/continuous-deployment), which focuses on automating releases of projects as soon as possible.

- Operate for conducting the development operations of configuration management and continuous monitoring.

#### DEVOPS PRINCIPLE

At the heart of DevOps principles, you will find the idea of collaborative learning and collaborative relationships between development and operations. They focus on increasing the pace of planned work for higher deployment rates, while also upgrading the reliability, stability, resilience, and security of the production environment. To establish an organization based on DevOps principles, you need to emphasize this holistic, whole-system approach across not just the development and operations departments but also every surrounding department and support organization within the company. In return, your whole system should be used to shape your organizational goals.

- Process improvement initiatives to truncate feedback loops to continuously implement needed bug fixes and vulnerability remediation earlier and more cost effectively

- Continual experimentation that encourages risk-taking and learning from success and failure, so continuous attempts will lead to future success and mastery

- Learners becoming teachers and passing along their acquired knowledge to their colleagues

- Using DevOps automation to improve efficiency

**DevOps in the Real World**

The software and IT industries are about speed and efficiency. DevOps has emerged as a paradigm to bring innovative products and features faster to the market. Many technologies have recently popped up to smooth the transition between development and operations. They have in common fluid delivery practices, thus managing complexity. Technologies for fast application delivery, such as for a search or selling platform, don’t apply to embedded software. But we can certainly learn from those approaches and map them to other domains. As we mentioned before, the emergence of over-the-air technologies for fast automotive software updates shows that the principles are transferable, while taking into account the stringent needs of continuous functional safety and security. Cloud and Web development have been early adopters of DevOps practices and can serve as guide for other domains. For example, Amazon Web Services (AWS) offers several tools for implementing continuous delivery. One such tool is AWS Elastic Beanstalk, which supports continuous deployment with an easy approach and therefore a shallower learning curve, but with less configurability. AWS Ops Works offers a middle way, in which you can code infrastructure; it offers integration with Chef. You can also create an AWS Cloud Formation template, written in a JSON format, to provision infrastructure in a repeatable way and to control all the cloud infrastructure components. You can use the AWS Code Deploy service to deploy applications across multiple virtual machines (Elastic Compute Cloud instances) with minimal downtime. Alternatively, you can use AWS Code Pipeline. This service, launched in 2015, integrates build, testing, and implementation. Complementarily, you can use other AWS components to support continuous delivery. AWS Code Commit is a managed source control service that hosts private repositories. AWS Cloud Watch offers a monitoring and alerting infrastructure that can help the whole team work on the deployed application’s reliability and performance. Figure 2 shows a DevOps architecture that uses AWS tools.

**ANSIBLE**

Ansible provides reliability, consistency, and scalability to your IT infrastructure. You can automate configurations of databases, storage, networks, firewalls using Ansible. It makes sure that all the necessary packages and all other software are consistent on the server to run the application. Let’s take an example; you’ve got a debug version of an application that is built on visual C++. Now if you want to run that application on a computer, you would need to meet some prerequisites like Microsoft Visual C++ library DLLs, and you would need visual C++ installed in your computer. So, this is the part where Ansible will make sure that all these basic packages and all the software’s are installed in your computer so that your application can run smoothly on all the environments, may it be test or production environment. It also holds all the historical data of your application, so if at any time you want to roll back to the previous version, or you want to upgrade it, you can easily do that. Let’s take a look at some of the following features.

**Agentless** – Which means there is no kind of software or any agent managing the node like other solution such as puppet and chef.

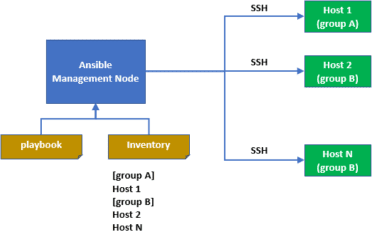
**Python** – Built on top of python, which is fast and one of the robust programming languages in today’s world.

**SSH** – Very simple passwordless network authentication protocol which is secure. So, your responsibility is to copy this key to the client

**Push architecture** – Push the necessary configurations to them, clients. All you have to do is, write down those configurations (playbook) and push them all at once to the nodes. You see how powerful it can be to push the changes to thousands of servers in minutes.

**Setup** – a minimal requirement and configuration needed to get it to work.

#### ansible architecture - what is ansible ANSIBLE ARCHITECTURE

Let us start with Public/Private Cloud which is the Linux server. It can also act as a repository for all IT installation and configurations.

The above architecture has a bunch of **host** machines to which ansible server connects and pushes the playbooks through SSH. It has **ansible automation engine** using which users can directly run a playbook which gets deployed on the hosts. There are multiple components in the ansible automation engine. The first is a **host inventory**. It’s a list of all the IP addresses of all the hosts. Next, there are **modules**. Ansible comes with hundreds of inbuilt modules and modules are those pieces of code that get executed when you run a playbook. A playbook contains plays, a play contains different tasks, and a task includes modules.

When you run a playbook, it’s the modules that get executed on your hosts, and these modules contain action in them. So, when you run a playbook, those action takes place on your host machines. You can make your custom modules also. All you must do is write a few lines of code and make it your module, and you can run it anytime you want. Then the architecture has **playbooks**.

Playbooks here actually define your workflow because whatever tasks that you write in a playbook, it gets executed in the same order that you have written them. For example, if you have written that install a package first and then start, it’ll do the same. Playbooks are very simple to write YAML code.

YAML code is a very simple data serialization language; it’s pretty much like English. Next, in the architecture are plugins. Plugins here are special kind of modules.

These plugins get executed before a module is getting executed on the nodes. Plugins get executed on the main control machine for logging purposes. You’ve got call-back plugins because this enables you to hook into different ansible events for display and logging purposes. Cache plugins are used to keep a cache of facts to avoid costly fact-gathering operations. Ansible also has action plugins, which are front-end modules, and they can execute tasks on the controller machine before calling the modules themselves.

#### How Ansible Works?

Ansible works by connecting to nodes and pushing out small programs called as ansible modules. Ansible will executes these modules over SSH by default and then remove them when finished.

Ansible management node is the controlling node, which controls the entire execution of the Playbook. It’s the node from which you are running the installation, and the inventory file provides the list of the host where the modules need to be run. The management node makes ssh connection, and then it executes the modules on the host machines and installs the product. It removes the modules once they are installed. So that’s how ansible works.

**SYSTEM ANALYSIS**

The beginning of networked computing when deploying and managing servers reliably and efficiently has been a challenge. Previously, system administrators managed servers by hand, installing software, changing configurations, and administering services on individual servers.

As data centers grew, and hosted applications became more complex, administrators realized they couldn’t scale their manual systems management as fast as the applications they were enabling. It also hampered the velocity of the work of the developers since the development team was agile and releasing software frequently, but IT operations were spending more time configuring the systems. That’s why server provisioning and configuration management tools came to flourish.

Consider the tedious routine of administering a server fleet. We always need to keep updating, pushing changes, copying files on them etc. These tasks make things very complicated and time consuming.

**PROPOSED SYSTEM**

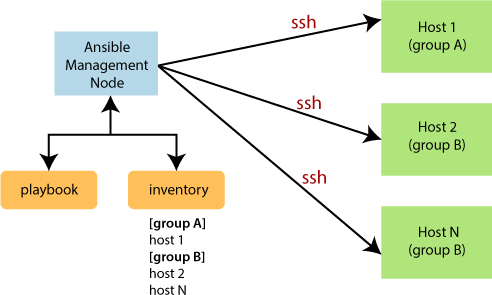
DevOps is a culture which promotes collaboration between Development and Operations Team to deploy code to production faster in an automated & repeatable way. The word 'DevOps' is a combination of two words 'development' and 'operations'. DevOps helps to increases an organization's speed to deliver applications and services. It allows organizations to serve their customers better and compete more strongly in the market.

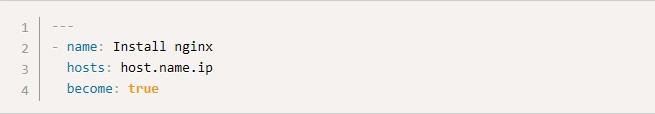
Ansible is one of the prominent DevOps tools for IT configuration management. Ansible is an open-source automation and configuration management platform by Red Hat. The prominent advantage of Ansible as a DevOps tool is in the use of YAML templates. YAML is a human-readable language and enables automatic repetitive programming tasks. The use of automated and repetitive processes rather than ad hoc scripting or manual configuration management ensures the effectiveness of Ansible.

**ANSIBLE WORKFLOW**

Ansible works by connecting to your nodes and pushing out a small program called **Ansible modules** to them. Then Ansible executed these modules and removed them after finished. The library of modules can reside on any machine, and there are no daemons, **servers,** or **databases** required.

In the above image, the **Management Node** is the controlling node that controls the entire execution of the playbook. The **inventory** file provides the list of hosts where the Ansible modules need to be run. The **Management Node** makes an **SSH** connection and executes the small modules on the host's machine and install the software.

Ansible removes the modules once those are installed so expertly.



**IMPLEMENTATION**

Ansible is a radically simple IT automation engine that automates [cloud](https://www.ansible.com/provisioning?hsLang=en-us) [provisioning,](https://www.ansible.com/provisioning?hsLang=en-us) [configuration management,](https://www.ansible.com/configuration-management?hsLang=en-us) [application deployment,](https://www.ansible.com/application-deployment?hsLang=en-us) [intra-service orchestration](https://www.ansible.com/orchestration?hsLang=en-us), and many other IT needs. **Playbooks** are essentially sets of instructions (plays) that you send to run on a single target or groups of targets (hosts). Think about the instructions you get for assembling an appliance or furniture. The manufacturer includes instructions so you can put the parts together in the correct order. When followed in order, the furniture looks like what was purchased.

HOW A PLAYBOOK WORKS

The Playbook we're building will install a web server on a target RHEL/CentOS 7 host, then write an index.html file based on a template file that will reside with the final Playbook.

Authors

-The author adds instructions for the modules to run, often with additional values (arguments, locations, etc.).

-The target host has modules run against it in the order the Playbook lays out (with includes or other additional files).

-The host's state is changed (or not) based on the results of the module running, which Ansible and Tower displays in output.

-With the furniture analogy, a Playbook is shorthand to tell the modules to perform a task. You must understand the following to run your Playbook successfully:

1. The target

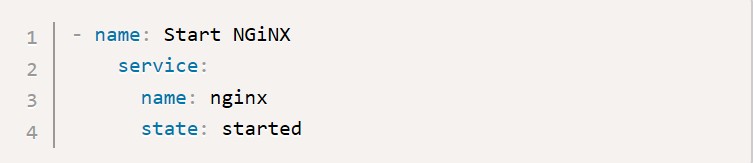
Because the Playbooks are providing direction and interactivity with the modules, Ansible assumes you know how to do what you're trying to do and automates it.

That's why Playbooks are like instructions or directions - you're telling the automated parts how you want the task configured.

2. The tasks

If part of the Playbook needs to start the **web server**, you're going to need to know how that's done so you know to use the service module and start the web server by name. If the Playbook is installing software, then you have to know how installation is done on the target.

Example Playbook

The target host will be a RHEL/CentOS 7 base install. There will be a web server installed (NGINX) and then an index.html file will be created in the default webroot. After the install and file tasks are completed the service will be started.

Playbooks start with the YAML three dashes (---) followed by:

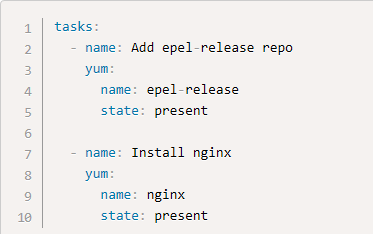
**Name:** good for keeping the Playbooks readable

**Hosts:**identifies the target for Ansible to run against

**Become statements**: true statement is included here to ensure nginx installs without a problem (it's not always required)

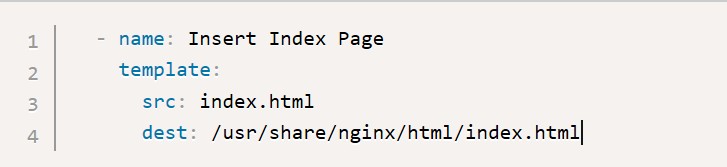
On the same indent level as the three prior statements will go the **tasks: statement**, after which any plays are listed another indent deeper (per YAML nesting). There are two tasks listed but both are using the Yum module. The first Yum task is adding the epel- release repo so that nginx can be installed. Once epel is present Yum is used to install the nginx package.

The state: present statement lets Ansible check the state on the target first before performing any further action. In both cases if the repo or package is already present, Ansible knows it doesn't have to do any more for this task and continues.

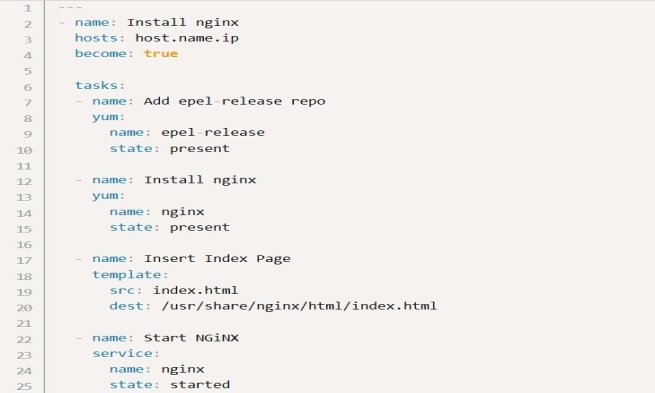


The default install page for nginx is fine if you want to test that nginx installed properly, but you might have a basic html file that you'd like to have as your confirmation. For simplicity, the template index file is in the same directory I'll run the Playbook from for the example.

The destination is simply the default for nginix with no configured sites.



The last thing the Playbook will do is make sure that the nginx service has been started (and if not, start it).

The entire Playbook is about the same length as the introduction paragraph:

CRONTAB

The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. Crontab stands for “cron table, ” because it uses the job scheduler cron to execute tasks; cron itself is named after “chronos, ” the Greek word for time.cron is the system process which will automatically perform tasks for you according to a set schedule. The schedule is called the crontab, which is also the name of the program used to edit that schedule.

Step 1: Install and Open Gnome Online Accounts

Ubuntu 18.04 usually comes with Gnome Online Accounts utility in the System Settings by default. In case your system lacks it, you can install it as follows:Open the Ubuntu command line, the Terminal, either through the system Dash or the Ctrl+Alt+T shortcut. Once the Terminal application opens, enter the following command as

**sudo:$ sudo apt install gnome-online-accounts**

Please remember that only an authorized user can add, removeand configure software on Ubuntu.

The above output shows that Gnome Online Accounts utility is already installed on my system. In case it is not installed, the system might prompt you with a y/n option to confirm initiating the installation procedure. Please enter Y and then hit Enter to continue.

Once the utility is installed on your system, you can open it through one of the following methods: By entering the following command in your Terminal:

**$gnome-control-center online-accounts**

Step 2: Add your Google Account to the Online Accounts

Once you have opened the Online Accounts utility, you will be able to see a list of online apps whose account you can configure to be used through Ubuntu. In our case, we want to access the Google Drive account so we need to add our Google account to the list of online accounts. To do so, click on the Google option from the list. Please make sure that you are connected to the internet. The following dialog will open for you to add your google account, Enter your Gmail ID on which you are using Google Drive and then hit Enter. This will open the view for you to enter the Password for this Gmail account.Enter your password and hit Enter. When the dialog will appear you will be able to view the list of permissions you are allowing GNOME, or your local system, over your Google Drive. Click the Allow button in order to give your Gnome access to your Google Drive. This will open the dialog for your list of items for which you want to use the online account. You can always come back and configure these lists of items. For now, please make sure that the Files slider button is turned on; only this way we will be able to access the Google Drive files through our Nautilus File Manager. Please close this dialog and you will now be able to go through your Google account added to the list of online accounts and close the Settings utility.

**Google Drive via google-drive-ocamlfuse PPA:**

Another alternative to a native GNOME Online Accounts feature is google-drive- ocamlfuse. google-drive-ocamlfuse is somewhat a free version of the paid OverGrive Google Drive GUI front-end.

In case from any reason the above access to your Google Drive fails using GNOME Online Accounts you can install google-drive-ocamlfuse by executing the below commands:

**$ sudo add-apt-repository ppa:alessandro-strada/ppa**

**$ sudo apt install google-drive-ocamlfuse**

Next, create a directory which you want to use to access your Google Drive files. For example, create google-drive directory within your user local directory:

**$ mkdir ~/google-drive**

At this stage, point google-drive-ocamlfuse to this newly created directory:

**$ google-drive-ocamlfuse**

**Schedule backup to Google drive for every 5 mins or hours or months:**

**Execute a cron job every 5 Minutes**

The first field is for Minutes. If you specify \* in this field, it runs every minutes. If you specify \*/5 in the 1st field, it runs every 5 minutes as shown below.

**\*/5 \* \* \* \* /home/folder/backup.sh**

Note: In the same way, use \*/10 for every 10 minutes, \*/15 for every 15 minutes, \*/30 for every 30 minutes, etc.

**Execute a cron job every 5 Hours**

The second field is for hours. If you specify \* in this field, it runs every hour. If you specify \*/5 in the 2nd field, it runs every 5 hours as shown below.

**0 \*/5 \* \* \* /home/folder/backup.sh**

**Execute a job every 5th weekday**

This example is not about scheduling “every 5 days”. But this is for scheduling “every 5th weekday”.

The 5th field is DOW (day of the week). If you specify \* in this field, it runs every day. To run every Friday, specify either 5 of Fri in this field.

The following example runs the backup.sh every Friday at midnight.

**0 0 \* \* 5 /home/ramesh/backup.sh**

(or)

**0 0 \* \* Fri /home/ramesh/backup.sh**

**Execute a job every 5 months**

There is no direct way of saying ‘every 5 months’, instead you have to specify what specific months you want to run the job. Probably you may want to run the job on 5th month (May), and 10th month (Oct).

The fourth field is for Months. If you specify \* in this field, it runs every month. To run for the specific month, you have to specify the number that corresponds to the month. For example, to run the job on May and Oct, you should specify 5,10 (or) you can simply use the 3 letter acronym of the month and specify May,Oct.

The third field is for DOM (Day of the Month). If you specify \* in this field, it runs every day of the month. If you specify 1 in this month, it runs 1st of the month.

The following example runs the backup.sh twice a year. i.e 1st May at midnight, and 1st Oct at midnight.

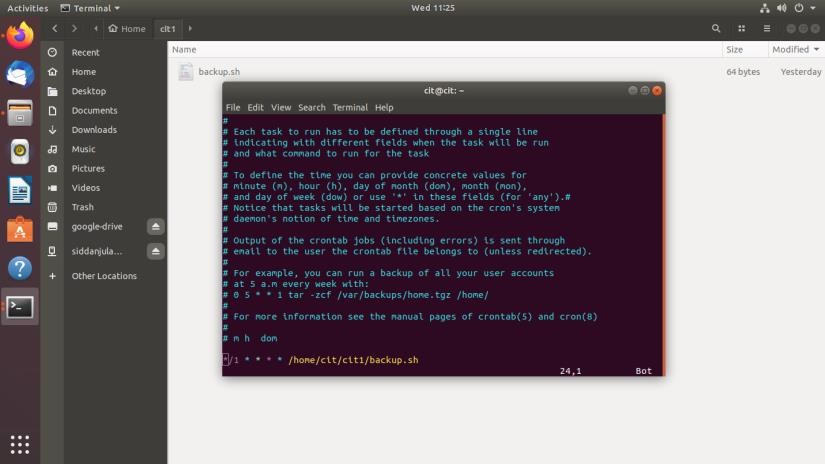
**0 0 1 5,10 \* /home/ramesh/backup.sh**

(or)

**0 0 1 May,Oct \* /home/ramesh/backup.sh**

**Code to backup for gdrive**:

**Mysqldump –u root -p cit > /home/cit/google-drive**

**Schedule backup: cronetab –e you will get the below window:**

**CONCLUSION**

Today, the infrastructure design is the software life-cycle phase that defines and configures the software infrastructure needs for that software as well as the number and type of physical hosts or virtual machines required. Infrastructure design typically consists of many installation and configuration scripts needed to, among others: (i) instantiate and link the required machines (either physical or virtual) for the software to run, (ii) install and configure the required software and middle-ware for those virtual machines, (iii) instantiate and run the needed ancillary services for the software to be operated. Following the current trends known as Infrastructure-asCode, we have created new layer for the Ansible framework which stands for the orchestration and configuration management framework. Newly created layer for Ansible offers management of university labs at Brno University of Technology, Czech Republic from local and public network.

Also, the new web interface has been created to simplify the configuration of tasks which are performed on daily basis – the created application is written in Java programming language utilizing the Spring framework. As the playbooks are designed to be human- readable, information about the operating systems are provided With the outlined goals accomplished, the management of machines from both the internal university network and public network was thoroughly tested via the web-interface and as an output, it can be stated all required software changes were successfully accommodated. The created framework represents a convenient tool enabling secured remote management and configuration of the selected network parts.

**REFERENCES**

[1] Elastic Compute Cloud(EC2), Amazon Web Services, Amazon. com, [http://aws.amazon.com/ec2.](http://aws.amazon.com/ec2)

[2] FutureGrid, https://portal.futuresystems.org. W. Gentzsch,Sun Grid Engine: towards creating a compute power grid, 1st IEEE/ACM Int. Symp.on Cluster Computing and the Grid (CCGrid 01), pp. 35 36, May 2001.

[3] G. Juve, E. Deelman, K. Vahi, and G. Mehta, Scientific Workflow Applications on Amazon EC2, 5th IEEE Int. Conf. on e-Science Workshop (e-Science 2009), pp. 59-66, Dec 2009.

[4] C. David, Introduction to DevOps on AWS, AWS, pp. 2, Dec. 2014 https://d0.awsstatic.com/whitepapers/AWS DevOps.pdf

[5] Sentinel Cloud Services, Run-time Guide, Safenet, 2014

[6] Sentinel Cloud Service, Sentinel Cloud V3.5 Quick Start Guide, Sentinel,Jun 2014

[7] Automated Provisioning and Deployment, Vmware White Papers,

[8] M. A. Murphy, B. Kagey, M. Fenn, and S. Goasguen, Dynamic Provisioning of Virtual Organization Clusters, 9th IEEE/ACM Int. Symp. On Cluster Computing and the Grid (CCGrid 09), pp. 364-371, May 2009. doi: 10.1109/CCGRID.2009.37

[9] Ansib[le, http://docs.ansible.com.](le,%20%20http://docs.ansible.com.)

[10] VLDB Endowment, vol.2, Isss.1, pp. 253-264, Aug. 2009.

[11] P. Shivam, A. Demberel, P. Gunda, D. Irwin, L. Grit, A. Yumerefendi,S. Babu, and J. Chase, Automated and On-Demand Provisioning of Virtual Machines for Database Applications, In ACM Proc. Int. Conf. Of Management of Data (SIGMOD 2007) pp. 1079- 1081, 2007