GC-EAD-DETAILS:

SSC-2003

INTER-2003-2005

BTECH-2005 TO 2009

US.ENTRY-OCT 2010 THROUGH H1B SPONSORED BY EGIANTS-CONVERGENT TECHNOLOGY

US FIRST PROJECT-Tesla Motors,Fremont,CA Feb 2016 – Till date,Sr.AWS/DevOps Engineer-domain: Automobile company

GC-EAD FILE DATE-11TH JULY 2011(MONDAY)

PRIORITY DATE FOR GC-EAD APPLIED DATE-11TH JULY 2011

LABOUR APP DATE:9TH FEB 2012(THURSDAY)

I-140 APP-APRIL 24TH 2013(TUESDAY)

H1B FOR 6YEARS(HOWEVER AFTER I-485 APPROVAL WE CAN EXTEND H1B MORE THAN 6YEARS.

EB2 CATEGORY SO BOTH H1 AND EB2 ARE VALID FOR 2020

I-485 APPLIED-NOV 29TH 2017(WEDNESDAY)

H1B-2010-2012

H1B 1ST EXTENSION- 2012-14

H1B 2ND EXTENSION-2014-2016

H1B 3RD EXTENSION-2016-2017(EB2-CATEGORY)

H1B 4TH EXTENSION-2017-2019(EB2-CATEGORY)

**PROJECTS:**

**Current project: Tesla Motors,Fremont,CA Feb 2016 – Till date,Sr.AWS/DevOps Engineer-domain:Automobile company**

**Client4: FIS global, Ann Arbor, MI ---------- Dec 2014 – Feb 2016 -Devops/Aws-engineer-domain-financial and professional services.**  
**Client3: US Bank, Florham ParkNJ - Duration: Sept 2012-Nov 2014 -Devops engineer**  
**Client2: Humana Health Insurance, Los Angeles,CA,Oct2010-Aug2012 -Build and release engineer.**   
**Food world groups -Bangalore,India- June 2009 – Sep 2010-linux admin**

**I will quickly talk about myself. I am santosh and i have 8+years of experience in IT industry.from past 5years i have been working as both aws/devops engineer,and also in the past I worked as a build and release engineer.I started my career as a linux administrator having a good skillsets in linux ie bash scripting and automation.**

**My current role is with Tesla Motors, Fremont, CA ,where in I am working as Sr.Devops/Aws Engineer with a team of 10members in devops managing 3product applications and all of them are related to java/j2ee based n-tier applications architecture. We used microservices written in spring boot on relational data base.**

**I was involved in designing and deploying web-based applications utilizing almost all of the aws stack,I also helped in migrating the current linux environment to AWS and lots of automation using python,bash scripting,terraform and cloud formations.**

**I also have experience with CI/CD pipelines specifically I have worked on jenkins ,Ansible,puppet,chef,docker familarly with docker swarm and kubernetes creating pods on different node clusters running many applications.**

|  |  |
| --- | --- |
| Build Tools | Ant, Maven, Gradle. |
| Configuration Management Tool | Chef, Ansible, Puppet and Terraform |
| CI/CD Tools | Jenkins/Hudson, Jenkins 2.0, Team city, Bamboo, Octopus. |
| Cloud Services | Amazon Web Services(AWS), Digital Ocean, Microsoft Azure |
| Software container | Docker. Docker swarm, Kubernetes, Vagrant, Micro services. |
| Monitoring Tool | Cloud watch, Splunk, Nagios, Nagios XI, ELK |
| Version Control System | Subversion(SVN), GitHub, Bitbucket, GitLab, Perforce, TFS, CVS. |
| Repositories | Nexus artifactory, Jfrog Artifactory, Anthill Pro |
| Programming Languages | C, Java/J2EE, python. |
| Scripting Languages | Bash, PowerShell, Unix Shell Scripting, Python, Groovy, Perl, Ruby. |
| Application/Web Servers | Tomcat, IIS, Web Logic, JBoss, WebSphere, Nginx. |
| Databases | Oracle, MY SQL, NoSQL, SQL Plus, Redis, Mongo DB, and Cassandra. |
| SDLC | Waterfall, Agile and Scrum methodologies |

Technical skills:

Build tools:

An **artifact** is a file, usually a JAR, that gets deployed to a **Maven** repository. A **Maven**build produces one or more **artifacts**, such as a compiled JAR and a "sources" JAR. Each **artifact** has a group ID (usually a reversed domain name, like com.example.foo), an **artifact** ID (just a name), and a version string.

**Artifact**. **Artifacts** can be used to represent data created as a side-effect of running a Jenkins build. **Artifacts** are files which are associated with a single build. A build can have any number of **artifacts** associated with it

.

A) Maven:**Maven** is a **build tool**, in short a successor of ant. It helps in **build** and version control. However **Jenkins** is continuous integration system, where in **maven** is **used** for **build**. **Jenkins** can be **used** to automate the deployment process.

1. Ant:Apache **Ant** is a Java library and command-line **tool** whose mission is to drive processes described in **build** files as targets and extension points dependent upon each other. The main known usage of **Ant** is the **build** of Java applications
2. In short, though **Maven** and **ANT** are build tool but main difference is that **maven** also provides dependency management, standard project layout and project management. On difference between **Maven**, **ANT** and Jenkins, later is a continuous integration tool which is much more than build tool
3. .Gradle:**Gradle** is an open-source **build** automation system that **builds** upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration.

Config mgmt tools:

A) chef:chef is an automation tool platform that configures and manages your infrastructure and transforms them into code. **Chef automates how infrastructure is configured, deployed, and managed across your network, no matter its size.**

· The Chef DK workstation is the location where users interact with Chef. On the workstation users author and test [cookbooks](https://docs.chef.io/cookbooks.html) using tools such as [Test Kitchen](https://docs.chef.io/kitchen.html) and interact with the Chef server using the [knife](https://docs.chef.io/knife.html) and [chef](https://docs.chef.io/ctl_chef.html)command line tools.

· Chef client nodes are the machines that are managed by Chef. The Chef client is installed on each node and is used to configure the node to its desired state.

· The Chef server acts as [a hub for configuration data](https://docs.chef.io/server_components.html). The Chef server stores cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by Chef. Nodes use the Chef client to ask the Chef server for configuration details, such as recipes, templates, and file distributions.

· Chef is a combination of a chef server,nodes & workstation.server manages the nodes that make up the infra.

· Chef-client is a program that runs on each node,knife is a command line tool that runs on each station.

**B) PUPPET:IS A FRAME WORK FOR SYSYTEM AUTOMATION AND A CONFIG MGMT TOOL** which manage user accounts,groups,files and directories,package mgmt,configuration,managing services,volume mgmt,task scheduling,cron jobs,system backups.Open source platform written in ruby language used to deploy,scale,test.track infrastructure as a code.(scale quickly).It is a client-server model in which agent or client is connected to master which generates a catalog with list of resources that clients have to apply locally and send sends the reports to the master or server and have to wait 30 mins for the next config.codes written in manifests or files with .pp extension.Resources are grouped in classes and config are arranged in modules.

Manifests:files containing config data and a piece of code written to perform a task.components are resources,files,templates,nodes and classes

Modules:consists of various directories which also includes manifests.It is a collection of various components and manifests(files,directories,templates,external facts).

Ansible:Ansible is a config management provisioned IT automation tool used to build,deploy and manage an applications

**Storage** area networks (SANs) and **network attached storage** (**NAS**) both provide networked **storage** solutions. A **NAS** is a single **storage** device that operates on data files, while a **SAN** is a local network of multiple devices.

The Network File System (**NFS**) is a way of mounting **Linux** discs/directories over a network. An **NFS** server can export one or more directories that can then be mounted on a remote **Linux** machine. Note, that if you need to mount a **Linux**filesystem on a Windows machine, you need to use Samba/CIFS instead.

**Scripts** are reusable. Basically, a **script** is a text file containing the statements that comprise a **Python** program. Once you have created the **script**, you can execute it over and over without having to retype it each time. **Scripts** are editable.

[Apache](https://en.wikipedia.org/wiki/Apache_Software_Foundation) **Groovy** is a Java-syntax-compatible [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [programming language](https://en.wikipedia.org/wiki/Programming_language) for the [Java platform](https://en.wikipedia.org/wiki/Java_(software_platform)). It is both a static and [dynamic](https://en.wikipedia.org/wiki/Dynamic_programming_language) language with features similar to those of [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk). It can be used as both a [programming language](https://en.wikipedia.org/wiki/Programming_language) and a [scripting language](https://en.wikipedia.org/wiki/Scripting_language) for the Java Platform, is compiled to [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) [bytecode](https://en.wikipedia.org/wiki/Bytecode), and interoperates seamlessly with other Java code and [libraries](https://en.wikipedia.org/wiki/Library_(computing)). Groovy uses a [curly-bracket syntax](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) similar to Java's. Groovy supports [closures](https://en.wikipedia.org/wiki/Closure_(computer_programming)), multiline strings, and expressions embedded in strings. Much of the Groovy's power lies in its [AST](https://en.wikipedia.org/wiki/Abstract_syntax_tree) transformations, triggered through annotations.

Most valid Java files are also valid Groovy files. Although the two languages are similar, Groovy code can be more compact, because it does not need all the elements that Java needs.[[13]](https://en.wikipedia.org/wiki/Groovy_(programming_language)#cite_note-konig32-13) This makes it possible for Java programmers to learn Groovy gradually by starting with familiar Java syntax before acquiring more Groovy .

Groovy features not available in Java include both static and [dynamic](https://en.wikipedia.org/wiki/Type_system#DYNAMIC) typing (with the keyword def), [operator overloading](https://en.wikipedia.org/wiki/Operator_overloading), native syntax for lists and [associative arrays](https://en.wikipedia.org/wiki/Associative_array) (maps), native support for [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), [polymorphic iteration](https://en.wikipedia.org/w/index.php?title=Polymorphic_iteration&action=edit&redlink=1), expressions embedded inside strings, added helper methods, and the [safe navigation operator](https://en.wikipedia.org/wiki/Safe_navigation_operator) ?. to check automatically for [null pointers](https://en.wikipedia.org/wiki/Null_pointer) (for example, variable?.method(), or variable?.field).[[15]](https://en.wikipedia.org/wiki/Groovy_(programming_language)#cite_note-15)

Since version 2 Groovy also supports modularity (being able to ship only the needed jars according to the project needs, thus reducing the size of Groovy's library), type checking, static compiling, Project Coin syntax enhancements, [multicatch blocks](https://en.wikipedia.org/wiki/Java_syntax#Code_blocks) and ongoing performance enhancements using [JDK7](https://en.wikipedia.org/wiki/JDK7)'s invoke dynamic instruction.[[16]](https://en.wikipedia.org/wiki/Groovy_(programming_language)#cite_note-16)

Groovy provides native support for various [markup languages](https://en.wikipedia.org/wiki/Markup_language) such as [XML](https://en.wikipedia.org/wiki/XML) and [HTML](https://en.wikipedia.org/wiki/HTML), accomplished via an inline [Document Object Model](https://en.wikipedia.org/wiki/Document_Object_Model) (DOM) syntax. This feature enables the definition and manipulation of many types of heterogeneous data assets with a uniform and concise syntax and programming methodology.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)*]*

Unlike Java, a Groovy source code file can be executed as an (uncompiled) [script](https://en.wikipedia.org/wiki/Scripting_language), if it contains code outside any class definition, if it is a class with a *main* method, or if it is a *Runnable* or *GroovyTestCase*. A Groovy script is fully parsed, compiled, and generated before executing (similar to Perl and Ruby). This occurs under the hood, and the compiled version is not saved as an artifact of the process.[[17]](https://en.wikipedia.org/wiki/Groovy_(programming_language)#cite_note-konig37_8-17)

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a [relational database](https://aws.amazon.com/relational-database/) in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

Amazon RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including [Amazon Aurora](https://aws.amazon.com/rds/aurora/), [PostgreSQL](https://aws.amazon.com/rds/postgresql/), [MySQL](https://aws.amazon.com/rds/mysql/), [MariaDB](https://aws.amazon.com/rds/mariadb/), [Oracle](https://aws.amazon.com/rds/oracle/), and [Microsoft SQL Server](https://aws.amazon.com/rds/sqlserver/). You can use the [AWS Database Migration Service](https://aws.amazon.com/dms/) to easily migrate or replicate your existing databases to Amazon RDS.

1. Python Basics and Flow Control
2. Python Functions
3. Lists, Dictionaries, and Structuring Data
4. String manipulation and Regular expressions
5. Working with Files
6. Exception handling:In order to **handle** errors, you can set up **exception handling** blocks in your code. The keywords try and except are used to **catch exceptions**. When an **error** occurs within the try block, **Python** looks for a matching except block to **handle** it.
7. Debugging python scripts
8. Working with Linux system configuration.
9. Monitoring alerts using python
10. System health checks using python
11. Managing cron using python.

## Hosted monitoring services

Hosted monitoring software takes away the burden of deploying and operating the software yourself. However, hosted monitoring costs (often a significant amount of) money and take your application's data out of your hands so these services are not the right fit for every project.

Error Tracking

* [Rollbar](https://rollbar.com/) instruments both the server side and client side to capture and report exceptions. The [pyrollbar](https://rollbar.com/docs/notifier/pyrollbar/) code library provides quick integration for Python web applications. There are also specific instructions for common [web frameworks](https://www.fullstackpython.com/web-frameworks.html) such as [Django](https://www.fullstackpython.com/django.html) and [Pyramid](https://www.fullstackpython.com/pyramid.html).
* [Sentry](https://sentry.io/welcome/) is the hosted version of the open source tool that is used to monetize and support further development.

Application Performance Monitoring (APM)

* [New Relic](http://newrelic.com/) provides application and database monitoring as well as plug ins for capturing and analyzing data about other devleoper tools in your stack, such as [Twilio](https://www.fullstackpython.com/twilio.html).
* [Opbeat](https://opbeat.com/) Built for django. Opbeat combines performance metrics, release tracking, and error logging into a single simple service.
* [Scout](https://scoutapp.com/python-monitoring) monitors the performance of Django and Flask apps, auto-instrumenting views, SQL queries, templates, and more.

Status Pages

* [Status.io](http://status.io/) focuses on uptime and response metrics transparency for web applications.
* [StatusPage.io](https://www.statuspage.io/) (yes, there's both a Status and StatusPage.io) provides easy set up status pages for monitoring application up time.

Incident Management

* [PagerDuty](http://www.pagerduty.com/) alerts a designated person or group if there are stability, performance, or uptime issues with an application.

## Monitoring resources

* [How to Add Hosted Monitoring to Flask Web Applications](https://www.fullstackpython.com/blog/hosted-monitoring-flask-web-apps.html) and [How to Monitor Bottle Web Applications](https://www.fullstackpython.com/blog/monitor-python-web-applications.html) are a couple of posts in a series showing how to add hosted monitoring to Python web apps built with any of the major Python [web frameworks](https://www.fullstackpython.com/web-frameworks.html).
* [The Virtues of Monitoring](http://www.paperplanes.de/2011/1/5/the_virtues_of_monitoring.html)
* [Effortless Monitoring with collectd, Graphite, and Docker](http://blog.docker.io/2013/07/effortless-monitoring-with-collectd-graphite-and-docker/)
* [Practical Guide to StatsD/Graphite Monitoring](http://matt.aimonetti.net/posts/2013/06/26/practical-guide-to-graphite-monitoring/) is a detailed guide with code examples for monitoring infrastructure.
* Bit.ly describes the "[10 Things They Forgot to Monitor](http://word.bitly.com/post/74839060954/ten-things-to-monitor)" beyond the standard metrics such as disk & memory usage.
* [Four Linux server monitoring tools](http://aarvik.dk/four-linux-server-monitoring-and-management-tools/)
* [How to design useful monitoring and graphing visualizations](https://blog.serverdensity.com/how-to-design-useful-monitoring-graphs-and-visualizations/)
* [5 years of metrics and monitoring](https://speakerdeck.com/auxesis/5-years-of-metrics-and-monitoring) is a great presentation highlighting that visualization so humans can understand measurements is a hard problem. Line graphs are often not the best solution and they are overused.
* The Collector Highlight Series has an article on [StatsD](http://blog.librato.com/posts/statsd) that explains how to install it and how it works.
* This [survey on monitoring tools](http://kartar.net/2014/11/monitoring-survey---tools/) has some nice data and graphs on what developers and operations folks use in their environments.
* Ryan Frantz wrote a nice post on [Solving Monitoring](http://ryanfrantz.com/posts/solving-monitoring/) with a new definition of what monitoring means based on today's complex systems and how the practice should evolve going forward.

## Monitoring learning checklist

1. Review the software-as-a-service and open source monitoring tools below. Third party services tend to be easier to set up and host the data for you. Open source projects give you more control but you'll need to have additional servers ready for the monitoring.
2. My recommendation is to install [New Relic](http://newrelic.com/)'s free option with the trial period to see how it works with your app. It'll give you a good idea of the capabilities for application-level monitoring tools.
3. As your app scales take a look at setting up one of the the open source monitoring projects such as StatsD with Graphite. The combination of those two projects will give you fine-grained control over the system metrics you're collecting and visualizing.