Paper 1828-2018

A “Big Red Button” for SAS® administrators: Myth or Reality?

Sergey Iglov, SASIT Limited

ABSTRACT

Over the years, many attempts have been made to develop a solution that can make life for SAS® Administrators easier. Some were more successful than others, though none reached the point at which the process is fully automated, simple, efficient, fault-tolerant, and manageable. This paper explores the subject of so called Big Red Button—one solution that can make life for SAS Administrators as easy as clicking one button.

Starting with software provisioning through installation and configuration to support and maintenance, this paper looks at main steps of the SAS® Platform lifecycle, highlighting challenges and trying to find optimal solutions that link them all together. With virtualisation, automation, and cloud providers on the rise, in recent years the possibility of creating such a solution has finally become a reality. Automation frameworks allow for unattended, easily scalable installation processes, cloud based deployment provides an expandable and manageable infrastructure, and virtualisation helps to deliver software to clients through various channels.

All these glued together with orchestration tools form the basis for a potential Big Red Button. So, is it still a myth or can it be done now?

INTRODUCTION

Within a relatively short period of time the meaning of the word automation when used in the context of SAS® deployments has “exploded”.

A typical SAS Platform lifecycle contains 4 main stages: infrastructure provisioning; software installation and configuration; maintenance, and decommissioning. While elements of automation have existed at all four stages for a long time now, for example an unattended SAS® installation is nothing new, it wasn’t possible to glue all these steps together to provide a seamless end-to-end process that could be stretched across all stages.

With the new generation of SAS Software, also known as SAS® Viya®, wide spread of cloud platforms SAS Administrators finally got unprecedented opportunity to install, configure and maintain SAS® Platforms in the most efficient and almost fully automated manner.

Words like elastic, dynamically scalable, on-demand, flexible finally can be used when a SAS Platform installation, configuration and, what’s even more important, administration are mentioned.

Talking about “Big Red Button” it is important to explain what it actually means. If one goggles these three words together the most popular results will be:

- Killer switch;
- Emergency pull switch;
- Last resort action trigger.

It is clear that pressing this button requires a certain situation (almost always related to emergency or imminent critical failure), it also requires an assurance that by pressing it, actions triggered, will resolve either an on-going situation or prevent an upcoming problem. In short – these types of a buttons aren’t pressed often.

In many ways a lot of SAS Platforms once delivered gradually become “Big Red Buttons” that aren’t touched unless something serious has happened. It is fair enough – if a system works why bother touching it though what if business requirements have changed and a new capabilities are required; what if a critical update is released that has to be installed; what if an underlying infrastructure requires update or even replacement. All these situations will in one way of another lead to a “Big Red Button” being pressed. So why should it be that dangerous?

Before we look in details what is hidden behind a BRB (Big Red Button) it is important to set the scope as otherwise this paper can be easily hundred pages long due to the wide range of tool, potential scenarios and use cases. In the context of this paper we will be focusing on cloud deployment with the following subjects covered:

- Infrastructure provisioning
- Software installation
- Software configuration

A reasonable level of detail will be provided leaving low-level details out of scope. The main goal will be to answer the question – is it possible to create a process that is re-usable, adjustable and flexible without going mad?

All references to SAS® 9.x through this paper implies version 9.4M3 or later releases of SAS Software.

IAAS, PAAS OR SAAS – WHAT’S THE FUSS?

For many years prolonged lead times for the on-boarding, design and procurement of a new infrastructure had been the only option and considered normal. Nowadays though the cloud enables organizations to access new resources extremely fast. There is also a massive change from having to spend money under capital expenditure budgets to be able to meet demand throughout the lifecycle of an application or an infrastructure, which can quite often reach the order of magnitude of many years. A cloud deployment allows customers to begin with initial small size deployments and then scale infrastructure on-demand so that the size of it is always adequate to the workload placed on it. Among the most successful before cloud technologies arrived was virtualization. In some respect it was similar to what current cloud providers offer in a form of PaaS.

According to the National Institute of Standards and Technology (NIST), cloud computing is defined as: “a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

During the design phase one of the major decisions that an organization faces is whether to use a private or a public cloud. Both have pros and cons and decision to go either way is based on many factors like industry or government compliance requirements, business growth predictability, cost considerations etc. In the context of this paper we’ve chosen a public cloud offering as it gives us more flexibility and is readily available with generally lower than a private cloud cost.

Once type of a cloud deployment is known it is time to choose what cloud service model to use. Currently there are three options: Software as a Service (SaaS), Platform as a Service

Table 1 below shows common examples of IaaS, PaaS and SaaS

Platform type	Common examples
SaaS	Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, SAS® Cloud, SAS® Solutions on Demand
PaaS	Cloud Foundry, WS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift
IaaS	DigitalOcean, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

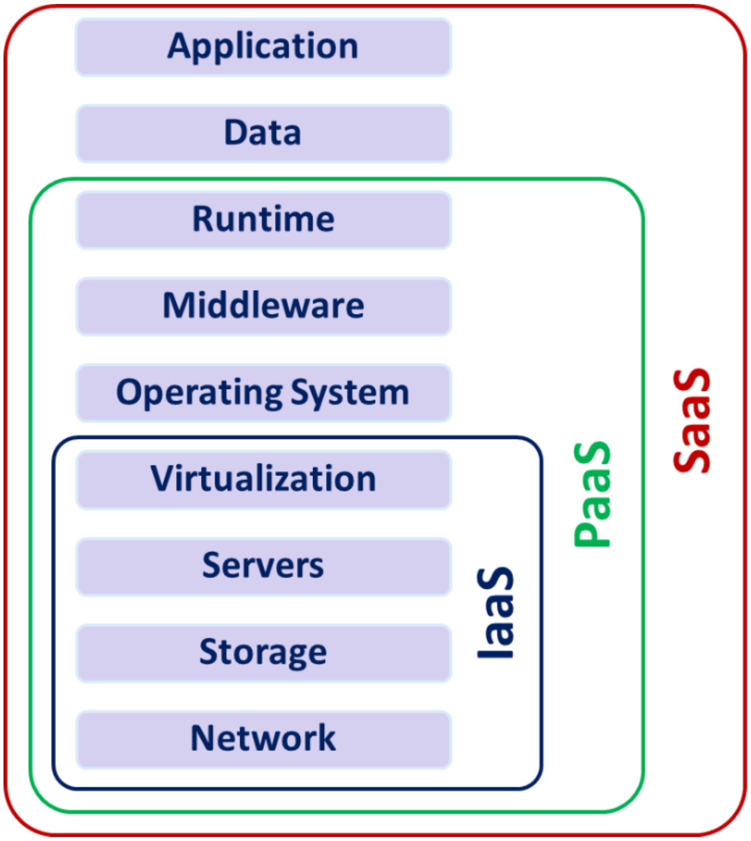


Figure 1. IaaS, PaaS and SaaS integration

First from the top is Software as a Service that is one of the most popular options for businesses world over. SaaS based applications are usually run directly through a web browser without any requirements for installation of client software. The task of managing software and its deployment is outsources to third-party services.

Platform as a Service typically provides a platform on which software components can be developed and deployed. It allows businesses to design and create applications with provided software components, not worrying about low-level infrastructure design and management.

This cloud computing service model provides more flexibility than SaaS and therefore allows organizations to build SAS Platforms that can meet their specific requirements and conditions while being scalable and highly available. SAS has announced last year that SAS Viya is now available on Cloud

Foundry and can be deployed using BOSH. Starting with SAS Viya 3.2 that is now shipped to the customers with a set of scripts it is possible to pull required BOSH releases that should be used by the BOSH Director service during a deployment. Releases available for both Ubuntu and CentOS (Red Hat) operating systems.

It is possible to complete full or programming-only SAS Viya 3.3 deployment using BOSH with an exception of Access to Hadoop configuration that can be taken care of by Ansible playbooks.

Once deployment is completed a number of manual steps should be completed out of which we found that configuring connection to authentication provide requires manual intervention the rest can be done using BOSH.

Using CF BOSH with the support from Ansible (for Embedded processes configuration mostly) it is possible to achieve the level of automation that can potentially allow almost 100% un-attended deployment of SAS Viya. It can be described with just a few commands:

```
bosh deployment SAS-Viya-manifest.yml

bosh deploy && bosh run errand microservices
```

First line will take YAML based BOSH manifest file that you've prepared before hands and set BOSH deployment to reference it. Second line will trigger software deployment. This is pretty much it. You just need to check that deployment was successful. You can validate SAS® Viya® deployment by running the following commands:

```
# display the instances in the current SAS® Viya® deployment.

      bosh instances

# verify that all of the SAS® Viya® microservices and applications are
deployed

      cf apps

# list IP addresses that are associated with the deployment,

      bosh vms
```

It should take around 20 min to stand up VDMML (SAS® Visual Data Mining and Machine learning) SMP (symmetric multiprocessing) test environment and around 35 minutes for MPP (massively parallel processing) deployment. Full deployment will take significantly longer. Obviously it is more to that that just a hand full of scripts and timelines mentioned above are for very simple deployments that aren't fit for Production where proper consideration should be given to security setup and a sizing.

Unlike SAS Viya that was built as cloud-enabled SAS 9.4 is only "cloud-ready". This means that it is possible to deploy SAS 9.4 onto a cloud based platform but the process of installation and configuration will be similar to a conventional deployment that starts with SAS Software depot being downloaded, pre-requisites fulfilled and finally SAS® Deployment Wizard used to install and configure SAS® software. A level of automation can be reached by using automation frameworks like Ansible, Puppet, Chef or Salt Stack so that deployment can be scripted and described as a code. Deploying SAS®9.4 software in the cloud does not make the SAS Software deployment process easier and it is worth remembering that:

- If there is a part of a platform that does not scale well, a cloud deployment would not make it more scalable. Quite often, scaling a SAS infrastructure results in an outage during the process.
- If a software deployment has not been designed for HA, a cloud deployment would not make it highly available;

- SAS Software installation process is the same in cloud as it is on premise.

Single machine deployment on SAS 9.4 EBI platform should take around 1 hour and majority of this time will be attributed to a silent installation and configuration with the help of prepared beforehand response file using SAS Deployment Wizard.

Simple deployment that delivers SAS® Studio as the front-end can be completed in less than 15 min either by using an automation framework mentioned above, Ansible playbook for instance, or by using containerized deployment.

Let's now move to the last cloud-computing model – Infrastructure as a Service. Located at the lower end of the managed cloud services spectrum IaaS provides highly scalable and automated compute resources. It is fully self-service with access to resources like compute, networking, storage, and other services, it also allows businesses to purchase resources as-needed instead of having to buy hardware outright.

Compare to PaaS or IaaS this cloud-computing model offers the most flexibility. It is literarily possible to design and built infrastructure from the ground up, starting with low level networking configuration up to monitoring, auto-scaling and service deployment. Combined with configuration management tools like Ansible, Puppet, chef and others infrastructure-provisioning step can now be seamlessly joined with software deployment creating end-to-end automated process.

FROM NOTHING TO SAS® VIYA®

INFRASTRUCTURE PROVISIONING

In the previous chapter we've looked across available cloud-computing models. Among 3 available IaaS allows the most flexibility and can cater for the most sophisticated requirements possible. In the context of this paper we'll be focusing on IaaS model and specifically on Amazon Web Services as the IaaS service provide.

As automation piece is written using CloudFormation (spoiler alert) that can be relatively easy migrated to Terraform to provide cross-cloud support the IaaS service provider of choice is AWS.

The same level automation can be achieved using other big IaaS service providers like Microsoft Azure, Google Cloud Compute, Rackspace etc.

Let's go back to AWS. It offers wide range of web services. We are going to use the following:

1. Compute - AWS' main offering is its EC2 instances, which can be tailored with a large number of options. There are three popular instance types when creating EC2 instances:
    a. Memory optimized. These instances are the most efficient considering per-GB memory cost.
    b. Compute optimizd. Ideal for instances that will require extremely high request rates;
    c. Storage optimized. Instances with extremely fast SSD storage.
2. Storage - Amazon EC2 offers flexible, cost effective, and easy-to-use data storage options for your instances. These storage options include the following:
    a. Amazon Elastic Block - behaves like a raw, unformatted, external block device that you can attach to a single instance;
    b. Amazon EC2 Instance – it provides temporary block-level storage for instances. It is deleted when an instance is stopped;

    c. Amazon Elastic File System (Amazon EFS) - provides scalable file storage for use with Amazon EC2;

    d. Amazon Simple Storage Service (Amazon S3) – provides access to reliable, scalable and inexpensive data storage infrastructure

3. Network – Among wide range of network and security features the most popular are:

    a. Amazon EC2 Key Pair - Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. The public and private keys are known as a *key pair*.

    b. Amazon EC2 Security Groups for Linux instances - A security group acts as a virtual firewall providing traffic controls for one or more instances;

    c. Amazon Virtual private cloud - it defines a virtual network in ring-fenced, logically isolated area within the AWS cloud, known as a virtual private cloud (VPC);

    d. Elastic IP - a static IPv4 address designed for dynamic cloud computing that allows access to a resources within VPC from the outside;

    e. Elastic Network Interface - represents a virtual network card that is a logical networking component of a VPC;

    f. Placement Groups – using placement groups helps to determines how instances are placed on underlying hardware as well as provide ways of organising similar resources;

AWS also provides related services such the EC2 Container service, AWS Lambda and Auto scaling.

As we'll see later using those components in relatively simple design allows us to provision infrastructure, deploy SAS Software and configure it (as part of the post-installation configuration) with one click of a mouse.

Figure 2 below shows infrastructure diagrams that describes an example infrastructure that will be deployed in one AWS Availability zone, will have it's own VPC with private subnet that contains security zone that governs access to EC2 instances (AWS calls EC2 resources - instances) deployed there as well as auto scaling group of EC2 instances. Information required for software installation and configuration as well as for EC2 instances pre-configuration is stored and access from S3 bucket.

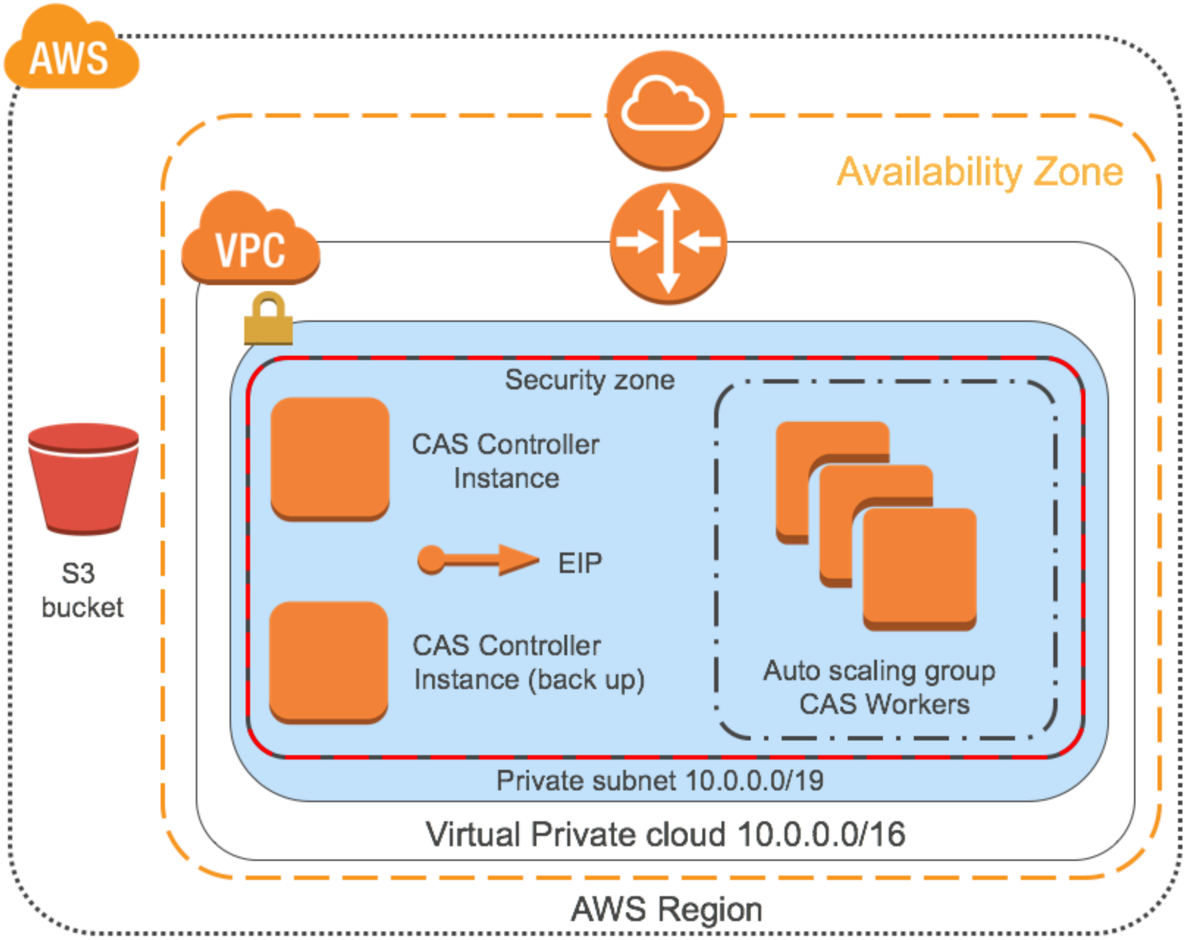

AWS
Availability Zone
VPC
Security zone
CAS Controller Instance
EIP
CAS Controller Instance (back up)
Auto scaling group CAS Workers
S3 bucket
Private subnet 10.0.0.0/19
Virtual Private cloud 10.0.0.0/16
AWS Region

Figure 2 – Example AWS based infrastructure for SAS® Viya®

Above diagram can be described using a number of CloudFormation templates. CloudFormation is free visual service that allows customers to create and manage the infrastructure they require to run a service (in our case SAS® Software) on Amazon Web Services. It has two parts: templates and stacks that give an easy way to create a collection of related AWS resources and provision them. A template can be presented in a form of a JavaScript Object Notation (JSON) or YAML Aren't Markup Language (YAML) text file. The stack is the set of AWS resources that are created and managed as a single unit when AWS CloudFormation deploys a template. CloudFormation automates and simplifies the task of creating groups of related resources in orderly and predictable way.

Below on figure 3 you can see an example of CloudFormation template that describes infrastructure required to run SAS Viya application. In addition to the standard supported CloudFormation resource types, it has custom resources that enable management of additional AWS and non-AWS resources. On the template shown on figure 3 below it is called "EC2Instances" and is linked with Lambda Function to support custom logic that is triggered anytime stake is created, updated (if you changed the custom resource), or deleted. In our situation custom logic allows us to select number of CAS worker nodes to be deployed (small – 2 or medium – 5).

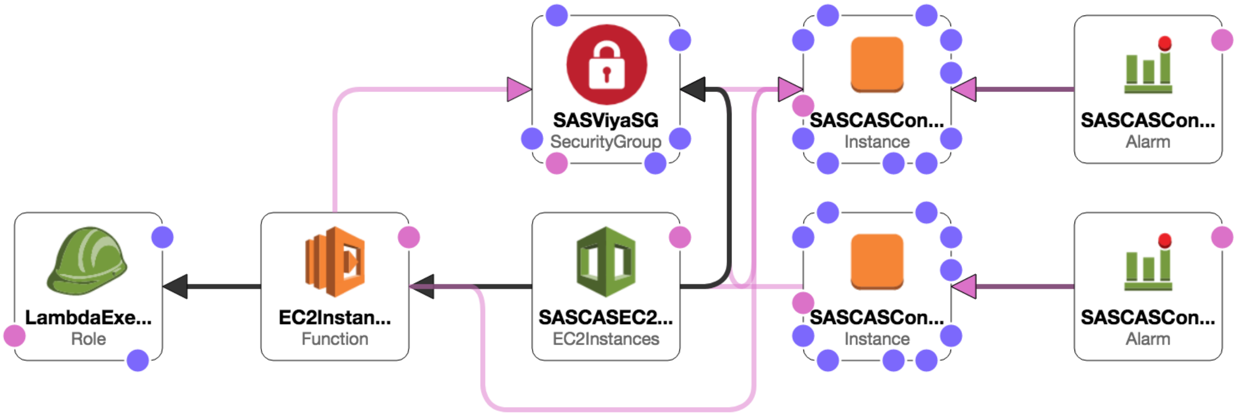
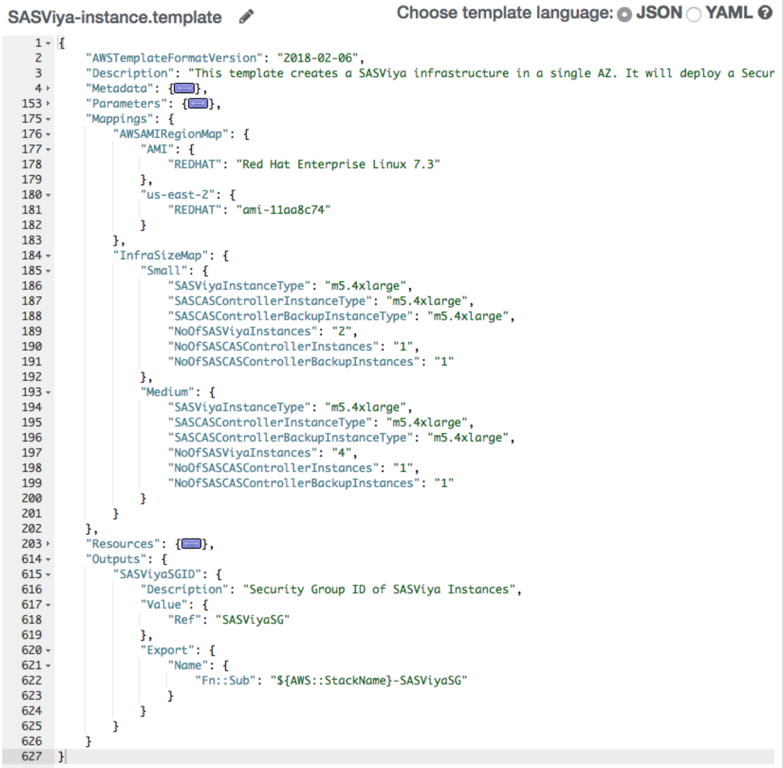


Figure 3 – Cloudformation template that describes infrastructure for SAS Viya deployment

Template above uses minimal number of an AWS resource types: IAM role, Lambda function, Custom Resource, Security group, EC2 Instance and CloudWatch Alarm. It creates an infrastructure in a single availability zone in a default VPC (though with minimal changes it can be enhanced to deliver its own separate VPC) with a default subnet. If there is a requirement to deliver a separate VPC it is recommended to have a look at public templates, especially at this one - https://github.com/aws-quickstart/quickstart-aws-vpc as a very good staring point. Dependencies between resources are presented with pink arrowed links that controls in what order certain template resources are created. The same diagram can be presented in a form of YAML (or JSON) document as shown below on figure 4.



SASViya-instance.template

Choose template language: ● JSON ○ YAML

```
{
    "AWSTemplateFormatVersion": "2018-02-06",
    "Description": "This template creates a SASViya infrastructure in a single AZ. It will deploy a Secur
    "Metadata": {↔},
    "Parameters": {↔},
    "Mappings": {
        "AWSAMIRegionMap": {
            "AMI": {
                "REDHAT": "Red Hat Enterprise Linux 7.3"
            },
            "us-east-2": {
                "REDHAT": "ami-11aa8c74"
            }
        },
        "InfraSizeMap": {
            "Small": {
                "SASViyaInstanceType": "m5.4xlarge",
                "SASCASControllerInstanceType": "m5.4xlarge",
                "SASCASControllerBackupInstanceType": "m5.4xlarge",
                "NoOfSASViyaInstances": "2",
                "NoOfSASCASControllerInstances": "1",
                "NoOfSASCASControllerBackupInstances": "1"
            },
            "Medium": {
                "SASViyaInstanceType": "m5.4xlarge",
                "SASCASControllerInstanceType": "m5.4xlarge",
                "SASCASControllerBackupInstanceType": "m5.4xlarge",
                "NoOfSASViyaInstances": "4",
                "NoOfSASCASControllerInstances": "1",
                "NoOfSASCASControllerBackupInstances": "1"
            }
        }
    },
    "Resources": {↔},
    "Outputs": {
        "SASViyaSGID": {
            "Description": "Security Group ID of SASViya Instances",
            "Value": {
                "Ref": "SASViyaSG"
            },
            "Export": {
                "Name": {
                    "Fn::Sub": "${AWS::StackName}-SASViyaSG"
                }
            }
        }
    }
}
```

Figure 4 – example of YAML based representation of a CloudFormation template

Once template is deployed using CloudFormation it can be used as a stack to simplify resources management. For example when a stack is deleted all related resources are deleted automatically as well.

To make it easier for customers to start using CloudFormation AWS published set of Quick Starts. They are reference deployments created by AWS architects together with partners (in our case SAS Software) that allow you to provision infrastructure that is required for specific software deployment out of the box. Quick Start for SAS® Grid Infrastructure can be found here. Simply open it in CF (CloudFormation) Designer, update or modify as required and deploy. It has all required components is certified by SAS Software and will provision an infrastructure that is ready for SAS® Grid deployment in a matter of minutes (please be aware that all resources created as the result of an execution of this Quick Start will attract cost). When status for a deployed stack changes to "Deploy Completed" it is time to install and configure SAS on the newly provisioned infrastructure.

SOFTWARE CONFIGURATION

In case of a SAS® 9.x deployment the process is identical to the one used in case of a conventional infrastructure delivery or an on premise deployment. SAS Deployment Wizard execution can be automated using response files, pre-requisites including SAS Software depot creation can be automated using a configuration management tool of choice. Post installation and configuration steps like external authentication provider configuration and some others can also be scripted and executed either on demand or automatically.

Starting with SAS Viya a new deployment tools are available to help with software installation and/or configuration process. In the context of this paper SAS Orchestration Command Line Interface (CLI) was used to generate set of Ansible playbooks that then were customized and used to automate deployment. Among generated playbooks there is one called system-assessment.yml that checks system pre-requisites and can be used to evaluate the state of a server and highlight pre-requisites that haven't been met. While it is possible to either adjust or write a new playbook from a scratch to perform all pre-requisites steps it is also possible to use one of the ready-made tools called – VIRK (SAS Viya Infrastructure Resource Kit). It can be cloned from the git repository and modified to address the needs of your specific deployment scenario. A combination of an example playbooks and VIRK allows customers to automate pre-deployment readiness tasks execution and validation.

One of the big differences in SAS Viya installation process compare to a SAS 9.x one is that it doesn't require a SAS Software depot to be created and maintained. It is recommended though to create and maintain a YUM Mirror Repository to ensure that the same of RMPs is used when deployments are separate in time. Software components are provided in the form of an RPM packages and installed using Linux utility tool – yum.

Depending on the number of SAS products licensed and included in an Order installation process for SAS Viya can take a long time to complete (comparable to similar software stack when SAS Deployment Wizard is used). Tt is recommended to plan an installation and configuration process, engage SAS and request an order that contains only SAS components that are required for a specific deployment to ensure that auto-generated playbooks have only components needed and all dependencies validated.

Another useful service that, in this case, AWS can offer is AMI (application management image). It provides the information required to launch an instance, which can be for example CAS worker and/or controller node. When created an AMI contains the following:

- A template for the root volume for an instance (for example, an operating system and application installation);

- Launch permissions that control which AWS accounts can use an AMI;
- A block device mapping that specifies the volumes to attach to the instance when it's launched (it is possible to share volumes across AMIs reducing maintenance and providing consistency, for example across all CAS worker nodes).

Using AMI can significantly reduce a platform deployment time as using pre-defined (or registered) AMI takes care of pre-requisites, installation and initial configuration. One of the use-cases where reduced deployment time can deliver improved user experience is a CAS worker scaling. Using one of the default playbooks called deploy-casworker.yml together with AMI it is possible to provision a new CAS workers dynamically based on the utilisation of a platform. Quick deployment together and no need to restart CAS controller allows building of a scalable (both out and in) platform that can deliver an uninterrupted service to business users.

Another way of linking Cloudformation template with Ansible based installation and configuration process is to use EC2 specific User data parameter. It allows passing a set of commands that is executed when as instance is launched (user data is not used when an instance is stopped or started after it was originally launched). User data is limited to 16Kb of raw data, has to be base64 encoded (EC2 instance can take care of encoding if provided data is not encoded before hands) and is treated on “what you give is what you get” basis. As an example, using user data it is possible to automate the following tasks:

- Mount EBS volume;
- Install required RPM packages;
- Configure Passwordless SSH access;
- Retrieve required data from an S3 bucket and use it to trigger Ansible playbook to configure SAS® software.

CONCLUSION

It is fair to say that – yes it is possible to create a re-usable, adjustable and flexible enough process of provisioning an infrastructure, installing and configuring software.. Modern technologies (configuration management tools), cloud based deployment and capabilities provided by cloud service providers makes it reality.

In this paper we used Amazon Web Services as the basis for creating an infrastructure for SAS Viya deployment. We used CloudFormation service to visualise the process of an infrastructure delivery. It also allows for repeatable and easily adjustable deployment of a designed infrastructure in a form of a stack. Stakes can be linked together to build a complex workflow.

We discuss how Ansible playbooks can automate all steps of SAS Platform delivery, from pre-requisites to validation and post-configuration changes. Example playbooks, available to all SAS customers even can automate scalability of a platform what can significantly reduce cost associated with running a platform and improve users experience.

REFERENCES

SAS® Viya® 3.3 Deployment Guide (Linux)

SAS® Viya® 3.3 Administration

SAS® 9.4 Intelligence Platform: System Administration Guide, Fourth Edition

SAS® 9.4 Intelligence Platform: Installation and Configuration Guide, Second Edition

Amazon Elastic Compute Cloud Documentation

AWS CloudFormation Documentation

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sergey Iglov

SASIT LIMITED,

London, UK

Phone: +44(0)7477-215-641

Sergey.Iglov@sasit.expert

Linked-in profile: linkedin.com/in/sergey-iglov