Paper 1354-2017

Transitioning Healthcare Analytic Platforms to the Cloud

Robert Lill, BNL Consulting and Jay Baker, BNL Consulting

ABSTRACT

As the IT industry moves to further embrace cloud computing and the benefits it enables, many companies have been slow to adopt these changes due to concern around data compliance. Compliance with state and federal law, and relevant regulations often leads decision makers to insist that systems dealing with protected health information or similarly sensitive data remain on-premises, as the risks for noncompliance are so high.

In this paper, we will detail BNL's standard practices for transitioning HIPAA compliant solutions from on-premises to a cloud based environment hosted by Amazon Web Services (AWS). We explain that by following best practices, and doing plenty of research, HIPAA compliance in a cloud environment is no more challenging than compliance in an on-premises environment.

We will review the role of best in-practice DevOps tools like Docker, Consul, ELK stack, and others which improve the reliability and the repeatability of your HIPAA compliant solutions. We will tie these recommendations to the use of common SAS™ tools in the “Analytics Results” section, and how they can work in concert in stabilize and improve the performance of the solution over the on-premises alternatives.

Although this paper will be focused on health care and HIPAA specific examples, many of the described practices and processes will apply to any sensitive data solutions that are being considered for the cloud.

INTRODUCTION

In today’s expanding healthcare analytics world, security has become more important than ever. Having a HIPAA compliant solution is almost a necessity for any healthcare IT solution. As the IT industry moves to further embrace "cloud" computing and the benefits it enables, many healthcare companies have been slow to adopt these technical improvements due to the inherent risks and compliance issues raised by the cloud paradigm. Primarily, that HIPAA compliance is hard enough to establish and maintain on-premises, so it must be significantly riskier and more difficult to maintain in a cloud environment.

Fortunately, this is not necessarily the case. Over the past several years, cloud providers, such as Amazon Web Services (AWS) have begun offering HIPAA compliant environments that, when combined with technical and business best practices, make meeting HIPAA compliance in a cloud environment is no more challenging than compliance on premises. Below, we will describe some of BNL’s related best practices and how BNL Consulting (BNL) has applied these to establishing and maintaining HIPAA environments for recent healthcare clients.

HEALTH INSURANCE PORTABILITY AND ACCOUNTABILITY ACT

HIPAA is a rather large bill, and provides rules and regulations for many facets of the healthcare industry: from medical records, to health insurance. Most often in the IT world though, we're primarily concerned with PHI: protected health information. PHI generally refers to any health information collected by a "Covered Entity" or a business associate of a covered entity. Covered entities refer to organizations that fall under HIPAA: hospitals, insurance companies, etc. Organizations that partner with covered entities

are business associates (in this case, both BNL's healthcare clients and BNL Consulting are business associates, and in turn Amazon Web Services). Both covered entities and business associates must comply with HIPAA. There are a lot of legal implications here, best left to your organization's legal or compliance teams.

Figure 1 System breaches

For a system administrator though, the most important aspects in HIPAA compliance will handle encryption, authentication, and authorization. Before you get to that stage though, do your homework. HIPAA violations and breaches can mean legal and financial trouble, both for your organization and yourself. There are lots of resources out there on this topic, but a great place to start is the AWS HIPAA Compliance White Paper. This is required reading for attempting a HIPAA compliant system on AWS.

CHECKLIST

Before you begin setting up what you hope is a HIPAA compliant environment in AWS, take some time to answer these questions that you might not have considered:

1. Do I even need HIPAA compliance?
   This one seems obvious, but unless you're certain to handle PHI at some point, then don't bother with HIPAA compliance. It's not terribly difficult, but it's also non-trivial.

2. Who will take charge of HIPAA compliance?
   There is a lot more to HIPAA compliance than Linux security hardening and encryption. Your organization needs someone other than a sysadmin to handle BAAs, HIPAA training, etc.

3. Do you have other internal resources that may be interacting with your new HIPAA compliant resources?

4. Unless you plan to convert all of your organization's internal resources to be HIPAA compliant, you will need to set clear boundaries between the two environments. This is fairly simple to do with AWS, but must be considered in advance.

GOTTA KEEP EM SEPARATED

There are a few key things to remember when designing and implementing your network topology for HIPAA compliance on AWS. As noted in question 3 above, if your organization has other resources that will not be HIPAA compliant on AWS they must be in a separate Virtual Private Cloud (VPC). You should

avoid joining your new HIPAA VPC via Virtual Private Network (VPN) to your other VPCs, or by other means.

When creating your HIPAA VPC, make the default instance type to be dedicated tenancy, as this is a requirement for HIPAA. That way, you won't accidentally forget to select that option when creating instances.

Your organization probably has many common ssh key pairs used when creating new instances. Create new ones for you HIPAA VPC, and distribute them only to those admins who will need them.

Another good practice is to create a logical separation of subnets within your VPC. Say your VPC encompasses a 10.0.0.0/16 subnet. You could then create multiple /24 subnets to separate out different roles your servers might take:

10.0.1.0/24 - Edge servers

10.0.2.0/24 - Admin servers

10.0.3.0/24 - Database servers

…

This makes it simple to create AWS security groups and firewall rules that are easy to understand and parse. For instance, if you are using MySQL, then your service layer needs to access the databases over port 3306 for MySQL, but it doesn't need ssh access. You can then create security group and firewall rules allowing traffic over port 3306 from your services subnet, but block port 22. And if you use OpenVPN as a means for your developers and testers to access your servers inside the VPC, then it makes it very easy to grant access only as needed. DBA can be granted access to the database subnet, but not the services subnet, and so on.

ENCRYPTION

The aspect of HIPAA most people first think of is encryption. HIPAA requires that any PHI be encrypted both "at rest" and "in transit". "At rest" refers to how the data is stored. You have a lot of options here: gpg file encryption, LUKS volume encryption coupled with LVM, or the simplest option is to use Amazon's EBS encryption. With encrypted EBS volumes, AWS handles the encryption for you. This greatly reduces the complexity of at rest encryption, allowing you to fulfill it with little more than a checkbox when creating volumes for you instances.

Still, it does come with some downsides. The data is unencrypted at mount time, meaning that once the encrypted volume is mounted to your instance, any user or application that has access to that part of the file system will be able to see your PHI. You can manage this problem with tools like SELinux and file permissions. You can also restrict ssh access to these servers (which you should be doing anyway). Another downside is that you can't encrypt the root volume of an instance at launch time when creating the instance. This means that it is important to clearly define what directories will be storing PHI, and ensure that your encrypted EBS volumes are mounted to those directories.

Using encrypted EBS volumes is a valid approach and the one most commonly adopted by BNL, but it is important to keep in mind the limitations. The bonus to this method is that you can layer gpg file encryption and/or LUKS with LVM on top of it for added security at the cost of added complexity.

Encryption "in transit" means that any time PHI data is sent over a network, it must be encrypted, even over the private network inside of an AWS Virtual Private Cloud (VPC). There are a few ways to tackle this. You could set up encryption for any services that might make PHI accessible over the network:

MySQL, Solr, MongoDB, etc. This method adds quite a bit of complexity, especially if you have multiple services which serve PHI (as is common), as each service will have different methods to achieve encryption. The solution most commonly employed by BNL is simply to encrypt all traffic between the servers. This does add some overhead, especially in the case of non-PHI data that must now be encrypted and decrypted as well, but we have found that overhead to be negligible, and the compliance assurance it brings is well worth the cost in any case. Below, we review some of the specifics of a recent migration for a BNL client.

TECHNOLOGY STACK

Below is the on-premises technology stack that BNL migrated to the cloud for the National Collaboration for BioPreparedness (NCBP):

- Red Hat Enterprise Linux 7.2
- Java for ETL
- Bespoke python for Analytics
- tc Server for Services layer
- AngularJS for UI
- Nginx Load Balancer and proxy

This technology stack was largely dedicated to a custom-built, web-based, analytics tool.

DEVOPS

Below is a list of the DevOps set of tools we employed for establishing and maintaining the new cloud environments. The motivation and benefits for implementing DevOps as part of BNL's best practices is probably beyond the scope of this topic, but simple Internet searching is well worth a few hours to understand how much this improves the stability and reliability of system administration.

- Ansible for config management and automation
- private git server, Jenkins automated builds, ELK Stack for central logging
- OSSEC for IDS, ClamAV for anti-virus
- tinc vpn, consul dns
- Docker for etl dev environment, possibly moving other parts of stack to docker

TINC & CONSUL: DISTRIBUTED VPN AND SERVICE DISCOVERY

As mentioned previously, we need to establish a VPN to ensure data in transit is encrypted. We chose Tinc.

Tinc is an open source (GPLv2) mesh VPN that supports a variety of encryption libraries and deployment configurations and that works by simple key exchange. If you are familiar with how to do ssh key exchange, then you're 90% of the way to setting up Tinc. You must exchange public keys between hosts in a simple config file which also includes an IP address for how to reach the host. Tinc then creates an encrypted tunnel to that host, which will appear as a separate interface. Any traffic sent over that interface will be encrypted in transit. Additionally, as there is no central server every node knows how to talk to every other node and will attempt to communicate directly to it, eliminating needless hops along the way.

Not to mention, the lack of a central server gives us high availability and fault tolerance: two traits that are highly desirable in a technology that's crucial to HIPAA compliance.

Once we have a VPN set up to encrypt all of our inter-node traffic, how do you get your developers to use it? You could always set up a DNS server that provides URLs to the Tinc IPs, but that's rather tedious. The tedium increases at a ridiculous pace as more servers and services are added. That's where consul comes in. Consul is tough to describe, as it does a lot in a small package. At its core though, it is a distributed service discovery tool and DNS server.

For our purposes, we run consul on each server (it deploys with a simple static binary), and have it bind on the Tinc vpn interface. Consul nodes communicate using a "gossip" protocol, which they use to share information about the health status of each node, what services it is advertising, and what its IP address is, etc. This information is made available both through Consul's easy to use http API, as well as through DNS, which is where consul really shines. All of our servers use consul for DNS. Because consul is distributed, every node can act as its own DNS server.

```
$ cat /etc/resolv.conf
nameserver 127.0.0.1
```

If you already have a DNS server in use, consul can recurse to it, making the transition seamless. Once all of the nodes are joined to consul, any node can be reached through consul's DNS feature at <host-name>.node.consul, like so:

```
$ ping jenkins.node.consul
PING jenkins.node.consul (10.0.0.1) 56(84) bytes of data.
64 bytes from jenkins.node.dc1.consul (10.0.0.1): icmp_seq=1 ttl=63
time=9.8 ms
...
```

Services can also be registered with consul, either through the consul HTTP API, or by defining them in configuration files, written in either JSON or HCL (Hashicorp Configuration Language). For example, you could register your UI at ui.service.consul, and have your load balancer point to that URL. Now you can freely move your UI to any server in your environment, so long as it's service config file travels with it. This extra layer of decoupling allows great flexibility, especially in a micro services or containerized environment.

Consul can also handle health checks, as well as act as a distributed key/value store ... but that's another blog post.

THAT'S GOING ON YOUR PERMANENT RECORD

Part of complying with HIPAA is logging any and all access to PHI. Some of this is handled by the web application that is the reason for this environment, but at the very least you will want to store and backup all system logs of any server in your environment. Merely storing these logs will probably cover the compliance aspect, but what if you're asked by a data provider to provide records of anyone who accessed your PHI? Or if you need to show which developers or admins were logged into a particular server during a certain time frame 6 months ago?

Centralized logging is a good practice for non-HIPAA systems for a variety of reasons, but for HIPAA compliant systems it's a must. We settled on the ELK (Elasticsearch, Logstash, and Kibana) stack to

solve this issue for us. All of the servers send their logs to a central server using Filebeat. Those logs are saved to Logstash, then indexed with Elasticsearch. Kibana serves as a flexible and easy to use web UI so that admins and developers can easily query system and application logs. With this setup, answering awkward questions from data providers or management is as simple as constructing a query in Kibana.

3RD PARTY AUDIT

To verify that our environment and practices complied with HIPAA, the NCBP project had a 3rd party company perform an audit of our AWS settings and our development practices. We provided them a read only IAM user so that they double check our AWS settings. They also conducted interviews with our team and even performed some penetration testing against our Web application.

Not only does this step help assure any stakeholders that you're complying with HIPAA guidelines, but it's better for a 3rd party auditor to discover problems with your system than a hacker. Even an internal audit, conducted by another admin at your organization is better than nothing. When dealing with HIPAA, non-compliance can be very expensive. It's best not to take chances. Spend the extra money to get someone to check over your work.

ANALYTIC RESULTS - THE EDWB (ENCOUNTER DATA WORK BENCH)

Another example of a HIPAA environment we managed at AWS for a client was the Encounter Data Workbench (EDWB). In this example, we worked with a SAS customer, the Centers for Medicare and Medicaid Services (CMS), to expand an application set to a HIPAA environment at AWS. Once this environment was established, the customer was able to expand their analytics platform and improve their ability to perform data analysis, ultimately viewing the data in many new ways, starting from the very broad and drilling to the very detailed.

At the cornerstone of the EDWB is SAS analytics. The visualizations as a result of the analytics are meant to convey relationships in the data where statistics are used to perform comparisons, determine compliance, highlight outliers, and easily identify patterns of change.

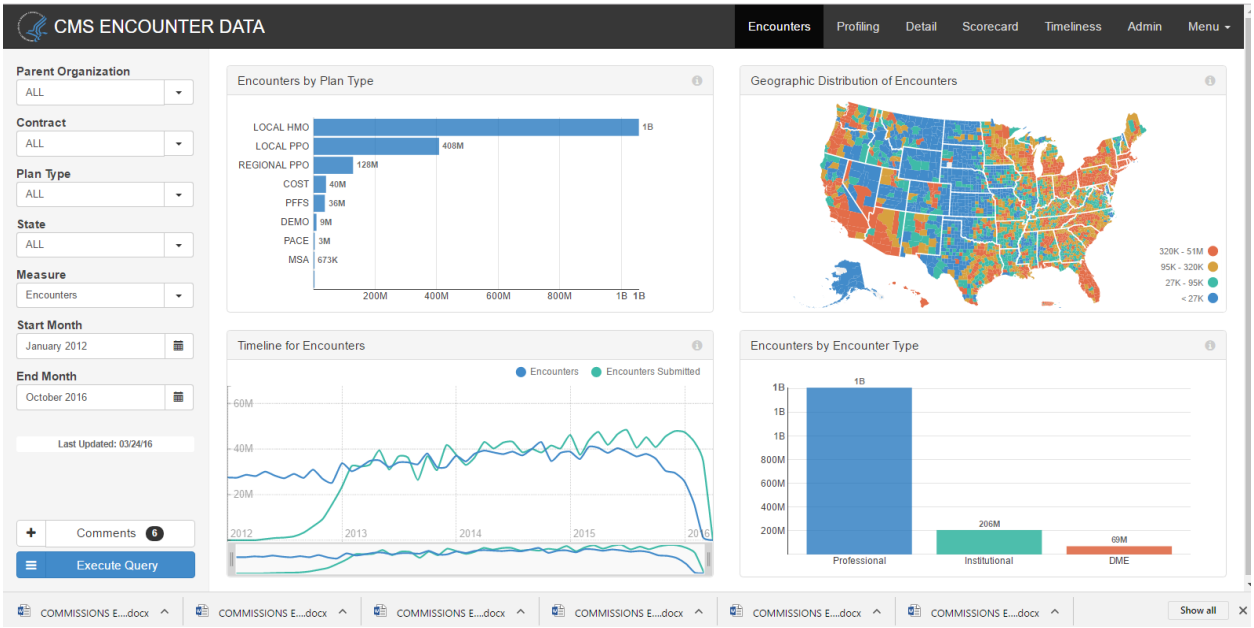

Figure 2 EDWB Encounters page

The architecture behind the scenes looks like the following. The architecture of this application is broken up into five tiers. This includes the client, mid, compute, VA and data tier. The following diagram shows the primary system components.

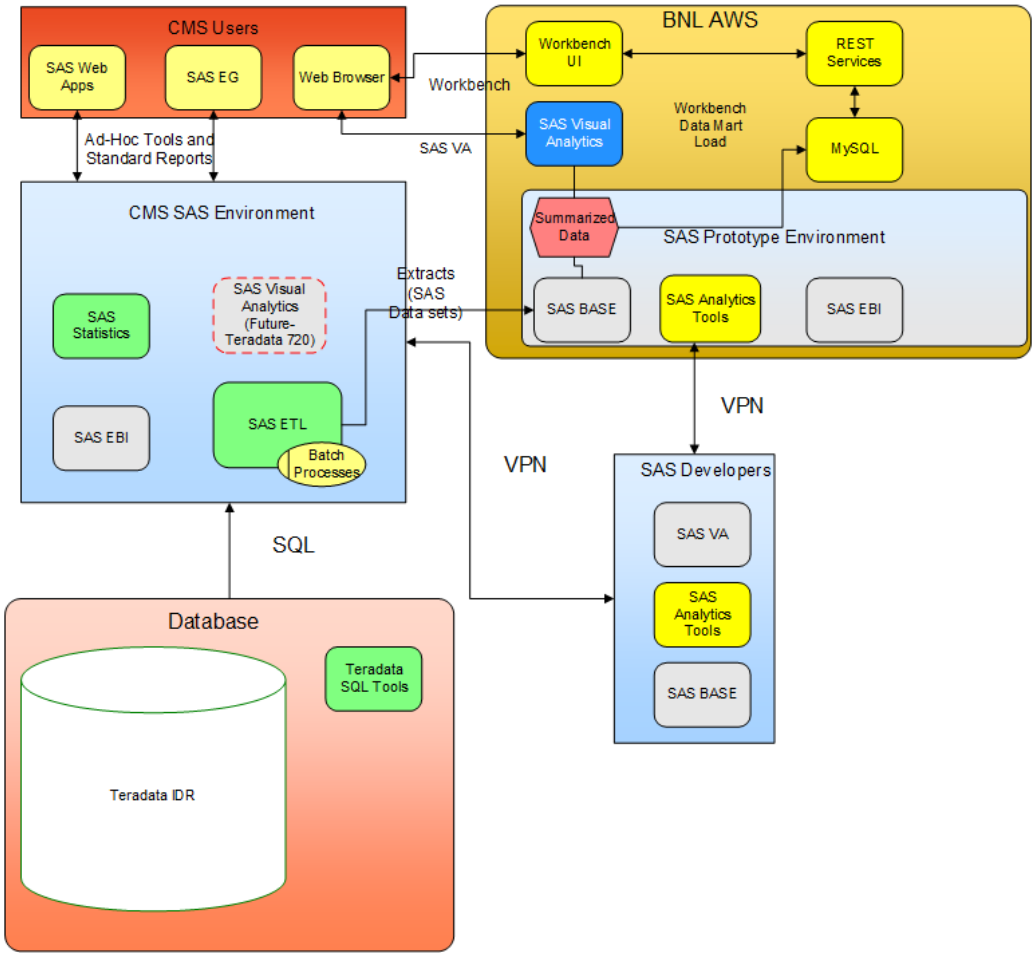

Figure 3 Original BNL and CMS Hosted EDWB architecture

As shown in the diagram, we have a mix of AWS and CMS servers (a hybrid approach). Most of the data preparation was done on CMS servers, while the data marts were built on the AWS servers. The data services and visualizations were served from the AWS environment as well.

This is also a distributed architecture that can support a variety of client types (web, Java, windows) connecting to the SAS Platform infrastructure. In this example, SAS is leveraged to provide report services, authentication services, and other metadata functions through the SAS Management Console.

This architecture is a good example of migrating SAS components from on-premises to a HIPAA compliant cloud. In this scenario, a SAS analytic system was expanded from an existing on-premises environment to a hybrid on-premises/cloud environment. This approach allowed the customer to make progress prior to determining hardware acquisition strategies or whether to migrate completely to the cloud.

A demo of the EDWB can be found at: https://secure.bnl-consulting.com/dev/cms/#/login. (Please contact BNL for a demo).

WHAT'S NEXT?

As new technologies mature and expand, BNL is working with SAS to make it easy for SAS customers to take advantage of new advances in cloud and DevOps technologies. For example, BNL is working with the new SAS Analytics for Containers package to make it simple for SAS developers to create temporary instances of SAS platforms that follow the container paradigm. This technology will make it possible for organizations to create SAS instances that are purpose built and, when used in collaboration with a scheduler or other automation tools, can be used to maximize the analytic computing power of on-premises environments, as well as cloud environments in an organization. Specialized instances that support HIPAA or another industry compliance standard will be easy to derive following the best practices described in this document.

CONCLUSION

HIPAA Compliance is an achievable standard for your environment. As we have shown in this paper, there are a number of technical steps, but none that can't be taken on, and with a 3rd party audit, you can get your environment approved. As described earlier, more and more organizations are going to ask for HIPAA level security even if they are not dealing with medical data.

CONTACT INFORMATION

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

Robert Lill
BNL Consulting
301-760-0709
rob.lill@bnl-consulting.com
http://bnlconsulting.com

Jay Baker
BNL Consulting
301-760-0709
jay.baker@bnl-consulting.com
http://bnlconsulting.com