AWS Cloud Best Practices

CloudBestPractices.net

Table of Contents

Netflix - Exemplar blueprint for Cloud Native Computing

The uber poster child of migrating legacy applications and IT systems via the 'Cloud Native' approach is Netflix. Not only do they share their best practices via blogs, they also share the software they've created to make it possible via open source.

Migrating to Web-Scale IT

In a VentureBeat article the author envisions 'the future of enterprise tech'.

They describe how pioneering organizations like Netflix are entirely embracing a Cloud paradigm for their business, moving away from the traditional approach of owning and operating your own data centre populated by EMC, Oracle and VMware.

Instead they are moving to 'web scale IT' via on demand rental of containers, commodity hardware and NoSQL databases, but critically it's not just about swapping out the infrastructure components.

Cloud Migration Best Practices

In **this blog** they focus on the migration of the core Netflix billing systems from their own data centre to AWS, and from Oracle to a Cassandra / MySQL combination, emphasizing in particular the scale and complexity of this database migration part of the Cloud Migration journey.

This inital quote from the Netflix blog sets the scene accordingly:

> *On January 4, 2016, right before Netflix expanded itself into 130 new countries, Netflix Billing infrastructure became 100% AWS cloud-native.*

They also reference a previous blog also describing this overall AWS journey, again quickly making the most incisive point - this time describing the primary inflection point in CIO decision making that this shift represents, a move to '**Web Scale IT**':

That is when we realized that we had to move away from vertically scaled single points of failure, like relational databases in our datacenter, towards highly reliable, horizontally scalable, distributed systems in the cloud.

Cloud Migration: Migrating Mission-critical Systems

They then go on to explain their experiences of a complex migration of highly sensitive, operational customer systems from their own data centre to AWS.

As you might imagine the core customer billing systems are the backbone of a digital delivery business like Netflix, handling everything from billing transactions through reporting feeds for SOX compliance, and face a 'change the tyre while the car is still moving' challenge of keeping front-facing systems available and consistent to ensure unbroken service for a globally expanding audience, while conducting a background process of migrating terabytes of data from on-site enterprise databases into the AWS service.

> *We had billions of rows of data, constantly changing and composed of all the historical data since Netflix's inception in 1997. It was growing every single minute in our large shared database on Oracle. To move all this data over to AWS, we needed to first transport and synchronize the data in real time, into a double digit Terabyte RDBMS in cloud.*
>
> *Being a SOX system added another layer of complexity, since all the migration and tooling needed to adhere to our SOX processes.*
>
> *Netflix was launching in many new countries and marching towards being global soon.*
>
> *Billing migration needed to happen without adversely impacting other teams that were busy with their own migration and global launch milestones."*

The scope of data migration and the real-time requirements highlight the challenging nature of Cloud Migrations, and how it goes far beyond a simple lift and shift of an application from one operating environment to another.

Database Modernization

The backbone of the challenge was how much code and data was interacting with Oracle, and so their goal was to 'disintegrate' that dependency into a services based architecture.

> *"Moving a database needs its own strategic planning:*

> *Database movement needs to be planned out while keeping the end goal in sight, or else it can go very wrong. There are many decisions to be made, from storage prediction to absorbing at least a year's worth of growth in data that translates into number of instances needed, licensing costs for both production and test environments, using RDS services vs. managing larger EC2 instances, ensuring that database architecture can address scalability, availability and reliability of data. Creating disaster recovery plan, planning minimal migration downtime possible and the list goes on. As part of this migration, we decided to migrate from licenced Oracle to open source MYSQL database running on Netflix managed EC2 instances."*

Overall this transformation scope and exercise included:

1. **APIs and Integrations -** The legacy billing systems ran via batch job updates, integrating messaging updates from services such as gift cards, and billing APIs are also fundamental to customer workflows such as signups, cancellations or address changes.
2. **Globalization -** Some of the APIs needed to be multi-region and highly available, so data was split into multiple Cassandra data stores. A data migration tool was written that transformed member billing attributes spread across many tables in oracle into a much smaller Cassandra structure.
3. **ACID -** Payment processing needed ACID transaction, and so was migrated to MySQL. Netflix worked with the AWS team to develop a multi-region, scalable architecture for their MySQL master with DRBD copy and multiple read replicas available in different regions, with toolingn and alerts for MySQL instances to ensure monitoring and recovery as needed.
4. **Data / Code Purging -** To optimize how much data needed migrated, the team conducted a review with business teams to identify what data was still actually live, and from that review purged many unnecessary and obsolete data sets. As part of this housekeeping obsolete code was also identified and removed.

A headline challenge was the real-time aspect, 'changing the tyre of the moving car', migrating data to MySQL that is constantly changing.

This was achieved through Oracle GoldenGate, which could replicate their tables across heterogeneous databases, along with ongoing incremental changes. It took a heavy testing period of two months to complete the migration via this approach.

Downtime Switchover

Downtime was needed for this scale of data migration, and to mitigate impact for users Netflix employed an approach of ***'decoupling user facing flows to shield customer experience from downtimes or other migration impacts'***.

All of their tooling was built around ability to migrate a country at a time and funnel traffic as needed. They worked with ecommerce and membership services to change integration in user workflows to an asynchronous model, building retry capabilities to rerun failed processing and repeat as needed.

An absolute requirement was SOX Compliance, and for this Netflix made use of components available in their OSS open source suite.

Our Cloud deployment tool Spinnaker was enhanced to capture details of deployment and pipe events to Chronos and our Big Data Platform for auditability. We needed to enhance Cassandra client for authentication and auditable actions. We wrote new alerts using Atlas that would help us in monitoring our applications and data in the Cloud.

Building HA, Globally Distributed Cloud Applications with AWS

Netflix provides a detailed, repeatable best practice case study for implementing AWS Cloud services, at an extremely large scale, and so is an ideal baseline candidate for any enterprise organization considering the same types of scale challenges, especially with an emphasis on HA - High Availability.

Two Netflix presentations: Globally Distributed Cloud Applications, and From Clouds to Roots provide a broad and deep review of their overall global architecture approach, in terms of exploiting AWS with the largest and most demanding of of capacity and growth requirements, such as hosting tens of thousands of virtual server instances to operate the Netflix service, auto-scaling by 3k/day.

This goes into a granular level of detail of how they monitor performance, and then additionally in they focus specifically on High Availability Architecture, providing a broad and deep blueprint for this scenario requirements.

Netflix Spinnaker - Global Continuous Delivery

In short these address the two core, common requirements of enterprise organizations, their global footprint and associated application hosting and content delivery requirements, and also their own software development practices - How better can they optimize the IT and innovation processes that deploys the software systems that needs this infrastructure.

Build Code Like Netflix - Continuous Deployment

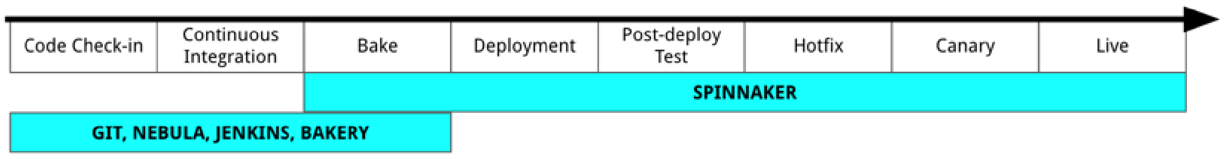
The ideal of our 'repo guide' for the Netflix OSS suite is for it to function as a 'recipe' for others to follow, ie You too can *Build Code Like Netflix*.

Therefore it's apt one of the best starting points is their blog with the same title - How We Build Code At Netflix.

Most notably because this introduces the role of Continuous Deployment best practices, and how one of their modules 'Spinnaker' is central to this.

Cloud Native Toolchain

In this blog Global Continuous Delivery With Spinnaker they explain how it addresses this scope of the code development lifecycle, across global teams, and forms the backbone of their DevOps 'toolchain', integrating with other tools such as Git, Nebula, Jenkins and Bakery.



As they describe:

Spinnaker is an open source multi-cloud Continuous Delivery platform for releasing software changes with high velocity and confidence. Spinnaker is designed with pluggability in mind; the platform aims to make it easy to extend and enhance cloud deployment models.

Their own quoted inspirations include Jez Humble's blog and book on Continuous Delivery, as well as experts such as Martin Fowler and working ideals such as 'Blue Green Deployments'.

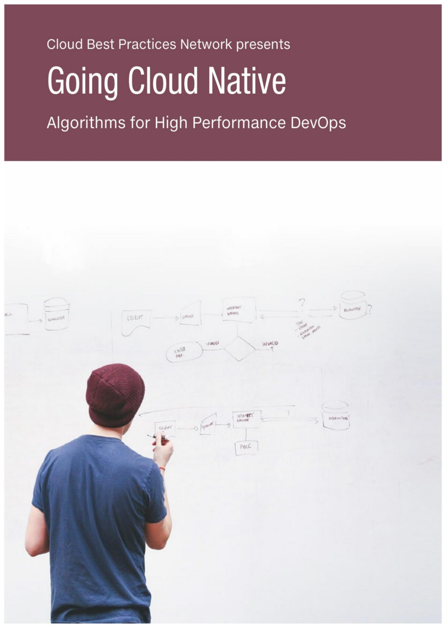
Moving from Asgard

Their history leading up to the conception and deployment of Spinnaker is helpful reading too; previously they utilized a tool called 'Asgard', and in Moving from Asgard:, describe the limitations they reached using that type of tool, and how instead they sought a new tool that could achieve:

- *"enable repeatable automated deployments captured as flexible pipelines and configurable pipeline stages*
- *provide a global view across all the environments that an application passes through in its deployment pipeline*
- *offer programmatic configuration and execution via a consistent and reliable API*
- *be easy to configure, maintain, and extend"*

These requirements formed into Spinnaker and the deployment practices they describe, which you can repeat through the Github Download.

Going Cloud Native with AWS


Cloud Best Practices Network presents
Going Cloud Native
Algorithms for High Performance DevOps

As the name suggests Cloud Native has become the term to describe software and systems that are purposely designed for the Cloud environments that will host them, versus the 'lift and shift' of traditional legacy applications.

Principly this means the use of containers for delivery, a microservices software architecture and Continuous Deployment practices for fast, regular modifications and upgrades to that code.

In this Going Cloud Native ebook (45 page PDF), I provide a general overview of the trend and also the transformation model, the 'algorithms', that can be used to design business practices that best exploit them.

AWS Cloud Native: Build Code Like Netflix

Naturally AWS is one of the primary Cloud providers for implementation of Cloud Native practices. As you would expect they support container implementation like Kubernetes and Docker.

It's also best to discuss tools for using them by also referencing their flagship customer case study - Netflix. Not only did they pioneer one of the core principles of microservices but they also open sourced the code they built to write, implement and manage them.

Therefore it's apt one of the best starting points is their blog with the same title - How We Build Code At Netflix.

Most notably because this introduces the role of Continuous Deployment best practices, and how one of their modules 'Spinnaker' is central to this.

Cloud Native Toolchain

In this blog Global Continuous Delivery With Spinnaker they explain how it addresses this scope of the code development lifecycle, across global teams, and forms the backbone of their DevOps 'toolchain', integrating with other tools such as Git, Nebula, Jenkins and Bakery.

As they describe:

> *Spinnaker is an open source multi-cloud Continuous Delivery platform for releasing software changes with high velocity and confidence. Spinnaker is designed with pluggability in mind; the platform aims to make it easy to extend and enhance cloud deployment models.*

Their own quoted inspirations include Jez Humble's blog and book on Continuous Delivery, as well as experts such as Martin Fowler and working ideals such as 'Blue Green Deployments'.

Continuous Deployment with AWS

So you can download these modules and literally code like Netflix, using AWS the same way they do, and also AWS offers Continuous Deployment capabilities as a set of Cloud services if you're looking for a simpler set up process and managed service.

CodeStar, CodeCommit, CodePipeline, CodeBuild and CodeDeploy offers a DevOps 'toolchain' for speeding the software development, build and deploy lifecycle. CodeDeploy also supports GitHub so that you can deploy application revisions stored in GitHub repositories or Amazon S3 buckets to instances.

AWS offers an excellent range of best practice white papers explaining the best practice use of the services, such as an Introduction to DevOps, Practicing Continuous Integration and Delivery on AWS, and using Jenkins on AWS, their dedicated blog offers regular insights, and this video offers guidance from one of their presentations, describing Cloud Native DevOps on AWS.

Partner solutions

It's also helpful to highlight some of the many partners in the AWS ecosystem who add value to this scenario.

For example BlazeMeter integrates with CodePipeline, adding load and performance testing into the pipeline process, enabling users to test APIs, mobile and web applications easily and rapidly.

Microservices on AWS

The third Cloud Native foundation component is a microservices software architecture and again there are a wealth of resources to learn from.

This 2016 AWS Summit presentation provides a comprehensive overview including the broader context of how it fits within this DevOps framework.

Their white paper provides a detailed review and this presentation dives more into the technical details and offers a number of implementation patterns:

Matias De Santi of Wolox describes how microservices can make use of AWS services like their API Gateway and RisingStack offers this article Deploying Node.js Microservices to AWS using Docker.

Microservices and Continuous Delivery

Nike's Journey to Microservices - Getting to Production Fast

Nike is also an organization that went through the Monolith to Microservice transformation, described in **this video** of their presentation at Amazon's 2014 Re:invent conference.

In this presentation he describes how the firm is committed to a fast pace of innovation and that digital is a top priority for the company, however upon starting his role he inherited a situation that proved contrary to these goals.

He recalls being presented with a 'code red' scenario, a production system that was barely stable, servers that wouldn't start, very little test and validation so that nearly every release contained defects, and despite engineers working 24/7 the new feature deployment cycle was months long.

At first they tried to tackle the situation by moving into a deep vertical stack using expensive technology, which tackled some issues but ultimately ended up 'building a monolith'. While this made many improvements including stabilizing the systems they still experienced issues like configuration management delaying the software cycle, database deployments were still too manual, and in general they were still moving too slowly, especially due to factors like slow organizational decision making.

Now they were down to a three week development cycle, but of which only two days was pure coding, the bulk was tied up in the stabilization processes. So Jason then began them on their journey towards a microservices approach.

Netflix OSS

They embraced the Phoenix pattern to implement immutable servers, via Amazon AMI instances, and adopted the shared nothing architecture. They also began making extensive use of the Netflix OSS stack, where tools Conformity Monkey combined very effectively with AWS features, like the management console and Amazon SNS & SQS for notification messaging, to provide them powerful architecture and tools, like dynamic discovery with Eureka.

Indeed their case study highlights just how powerful the combination of Cloud and microservices can be - For example at 25:00 he explains how "scaling concerns do not cross functional boundaries", meaning that capacity for one service can expand elastically without affecting any other services.

All of this was part of an overall goal of achieving a Continuous Delivery model, so that they could 'Get to Production Fast'. Now the delivery of new features can be achieved in days and weeks not months as their SCRUM team model works much more effectively, with more innovation projects possible while still also reducing the number of lines of code they work with, thanks to working with microservices not a big monolith.

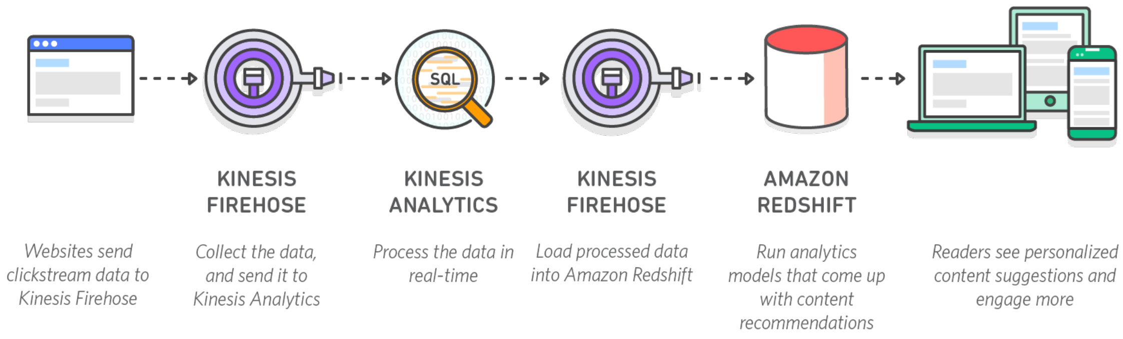
Building Smart Cities with AWS Kinesis

The Cloud accelerates exciting new innovations, making it easier for entrepreneurs to develop, build and launch new products much faster in hot sectors like the IoT (Internet of Things) and how it can be applied to scenarios like Smart Cities, as Amazon describe in their solution set and blog series.

From batch to real-time

An example of a key enabling Cloud service for this sector is AWS Kinesis, a suite of products that includes Firehose, Analytics, and Streams.

It's not a point solution specifically for Smart Cities rather a tool set that enables developers to build any type of application that processes large volumes of data streams, like sensor networks deployed around a city for monitoring all of its operations, from traffic tolls through garbage levels. As AWS describe this same capability could also be used for click streams from web sites.


SQL
KINESIS FIREHOSE
KINESIS ANALYTICS
KINESIS FIREHOSE
AMAZON REDSHIFT
Websites send clickstream data to Kinesis Firehose
Collect the data, and send it to Kinesis Analytics
Process the data in real-time
Load processed data into Amazon Redshift
Run analytics models that come up with content recommendations
Readers see personalized content suggestions and engage more

The key technological leap is from 'batch' processing, how IT architecture has traditionally dealt with handling very large volumes of data where it is stored up over time and then loaded into analytics tools in one job, to a real-time mode, where it is handled as it happens, as a stream of data.

The Zillow case study describes how the real estate business utilizes machine-learning calculation performance and scalability to deliver near-real-time home-valuation data to their customers. Netflix uses it for real-time monitoring of their vast infrastructure.

IoT Pragma Architecture - Small things and the Cloud

In their best practice white paper AWS provides a detailed overview of how to implement Kinesis, and in Core Tenets of IoT they define a 'Pragma Architecture' for building IoT solutions. This defines the component parts of building an overall IoT solution such as:

IoT Device Gateway: The interface between the devices and the processing applications, using protocols such as MQTT, WebSockets or HTTP.

Device Registry: A directory system for registering each of the devices and issuing them x.509 certificates, to enable secure communications via TLS.

Event Driven Apps: Middleware for enabling app integration, utilizing the SQS - Simple Queue Service and SNS - Simple Notification Service.

AWS Solution Architect Brett Francis gave this presentation that explains the detail of this blog, offering a primer for 'Small Things and the Cloud' and exploring the challenges of these telemetry applications.

Serverless Applications

IoT scenarios also provide an ideal context for 'Serverless' applications, an approach that can be implemented through the AWS Lamda service.

In essence it represents the pinnacle of Cloud computing because as the name suggests the key benefit is to abstract developers entirely from the underlying machinery of servers, freeing them to focus purely on adding value via the apps that they create.

It also works on an entirely utility model, Lamda executing code when it is needed and billing only for that consumption, ideal for the real-time, highly variable requirements of IoT scenarios, as AWS describe in this guide.

This video describes

Build your billion dollar SaaS venture on AWS - Best Practices

Become a SaaS Entrepreneur

When I pioneered my first SaaS venture back in 1994 it was on a DEC VMS mainframe and customers connected via private WAN links, and the whole kit and caboodle cost a few million bucks.

Now you have more computing power in your smartphone and you can rent an infinite capacity via the Internet, starting at free.

Given the valuations SaaS startups can sell for ranged from $2.3m through $9.3 BILLION for the top 10 deals in 2016, there's no other startup type that can consistently generate such a massive upside for such a low cost/risk entry.

With early 2017 deals like Cisco's $3.7 BILLION acquisition of AppDynamics the trend is only increasing exponentially, so what better time to launch your SaaS venture.

AWS for SaaS Startups

And you need look no further than AWS as your ideal platform partner. Not only do they offer all the raw computing power you'll ever need, they also offer a comprehensive best practices program specifically for SaaS startups.

Most notably this includes a blog category specifically on this topic and their SaaS Enablement Framework, an end-to-end aggregation of the common patterns and practices that are frequently used to build and deliver SaaS solutions on the AWS stack of services.

This earlier 2010 white paper SaaS on AWS provides a solid introduction to the general principles, and CloudAcademy offers a similarly helpful overview. AWS partner vendors like Blazeclan and Usersnap also provide helpful intros, from their tools perspective.

The AWS white paper Tenant Isolation Architectures offers a particularly meaty analysis of the core principles of SaaS design, and Rapid River explains how to master the related DNSsec challenge.

AWS also recently launched SaaS Subscriptions and one of their partners explores spanning your SaaS across multiple accounts.

Case studies

There are also awesome case studies available of SaaS ventures running on AWS:

- Encoding.com
- Market Simplified
- Jobvite

So what are you waiting for, the start of your billion dollar SaaS venture is only a click away.