FOGHORN WHITEPAPER SERIES

A Resource Guide For Navigating The Cloud

Container Empowerment On AWS

Harnessing the power of containers to transform application deployment and performance on AWS

aws partner network

Premier Consulting Partner

DevOps Competency

Security Competency

Solution Provider

foghornconsulting.com

In this whitepaper you will find out:

- > Introduction
- > Benefits of Containers
- > Anatomy of a Container
- > Securing Containers
- > AWS Tools for Container Success

Harnessing the power of containers to transform application deployment and performance on AWS.

INTRODUCTION

Back in the "ye old days"of 2012 when containers were just taking root in the DevOps ecosystem, most applications were monolithic. Within monolithic applications 100% of the communications within different functions of an application were internal to the application. As the application evolved and grew, so did its size and dependencies. With multiple developers working on different aspects of the application, it became harder and harder to get the code to build with zero errors. Builds began to take longer and longer. No matter the number of developers, the application's monolithic design meant that development velocity on a single app was limited.

With the advent of microservices, (whereby applications were subdivided into small, loosely coupled services based on functionality), teams could stay small and release quickly with less hassle. The drawback of microservices was that communication was now API driven, external to the application and each microservice needed its own infrastructure. All this agile overhead lead to overprovisioning (and overpaying) for infrastructure resources.

There was an appetite for a lighter weight, more dynamic system. How could development teams run multiple microservices on the same machine without them bumping into each other? With many of the apps talking and listening on the same port (https, port 443), what could be done? Enter containers and container orchestration.

Containers include all the OS and application libraries that a microservice needs in order to run. With an orchestration system like Kubernetes (K8s), developers can spin up lightweight containers in the place of servers and then let K8s know how many resources each container needs, and what port it wants to listen on. Based on POLICY, K8s can move containers around, translate ports, assign resources, check the health of the container, kill

31% of enterprises are currently using container technology, while another 29% are planning on using containers.

unhealthy containers by replacing them with new ones.

Containers are lightweight infrastructure components that enable deployment of applications easily by decoupling infrastructure dependencies from the application. Containers are a perfect infrastructure technology to enable a microservice application architecture.

Containers themselves are not new, but in the past several years containers have reached critical mass. As teams and engineers get more comfortable with container design and implementation this technology will continue to gain velocity. With their immutable and ephemeral nature, containers transcend physical locations running on prem and in the cloud without distinction.

Docker's open sourced Docker Engine launched in 2013, and has helped bring containers to critical mass. A recent Datamation Survey, titled State of the Cloud 2019 says that 31% of enterprises are currently using container technology, while another 29% are planning on using containers. According to Docker 100 million container images are downloaded per day. To illustrate the demand or containers, there are over 15,000 Linked In job postings for container expertise. It seems the only limiting factor for continued container growth in the cloud is limited by access to experience and knowledge of containers.

Containerized applications make sense for any organization putting agile methodologies into practice. Containers separate the application from the infrastructure, allowing it to run anywhere. In regards to DevSecOps, containers allow for end-to-edge security at scale. Automated security and governance are built into the application lifecycle, encouraging speed and an aerodynamic compliance ecosystem.

Innovation is driven by a human desire to maximize utilization, resources and performance. With the advent of DevOps and continuous integration and continuous development (CI/CD) innovation leaps have been the new expectation. Beginning with dedicated servers for applications, to VMs, containers are the natural evolution encouraging leaner, more agile applications that enable faster application testing, and scale and receive patches more elegantly and securely.

In this whitepaper we will explore the attributes, benefits and challenges of containerized microservices for development and deployment. We will explore container solutions as realized by the engineering brain trust of AWS engineers, and how these technologies can be adopted to increase performance and uptime while minimizing stress and cloud spend.

When we analyze the value of containers in relation to speed, the definition is threefold:

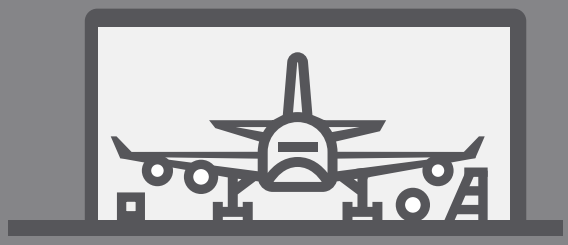
Lightweight containers, unburdened from the larger application, load substantially faster improving user experience.

For developers, containers increase innovation velocity by enabling CI/CD pipelines

For security protocols disposable containers & container orchestration make patching faster and config-management easier

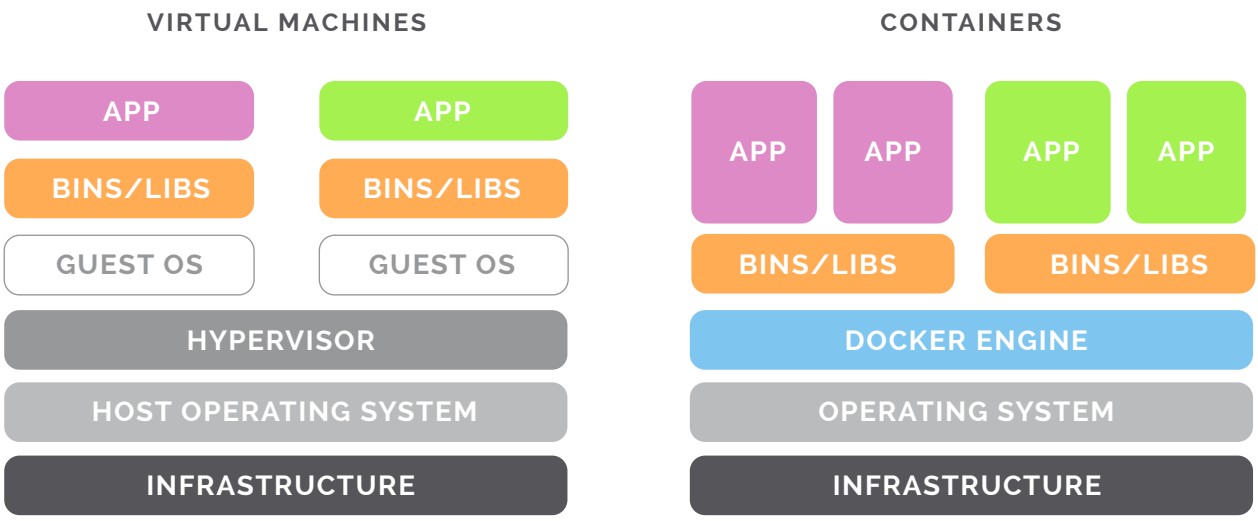
KEY BENEFITS OF A CONTAINER

Think of a monolithic application as a jet. With interconnected monolithic architecture the entire airplane must be taken to the hanger to have anything worked on. If there was a move to improve the air conditioning, or make the navigation more effective, the entire plane would be grounded. With binaries and libraries interconnected, the entire airplane (application) must be taken offline to be improved or debugged.



With containerization, an engine could be serviced mid flight. When through QA, containers allow for the air conditioning system to be swapped clicked into place like Lego. From development to deployment to user experience, containerization of a jet enables the "application" to work in a more seamless, agile, secure and nimble way that maximizes uptime.

To appreciate the advantages of Containers, we can compare them to another software deployment revolution that are Virtual Machines (VMs). VMs are a codified virtual computer, with its own virtual hardware, operating system, and networking. VMs and Containers share more in common than they differ. While they have different architectural frameworks, the goal of both Container and VM is to isolate an application and its dependencies into a self-contained unit that can run on any server. As well, Containers and VMs both have a private space for processing.



FLEXIBILITY

While VM's have their own OS, containers execute application processes in isolation. Just as a seafaring container on a ship can be placed upon different ships without losing any functionality, so can agile containers. The host OS therefore does not need specific software to run an application.

PLATFORM AGNOSTIC

Before containers applications were OS dependent, making it dependant on the hosting infrastructure. With configuration files and dependencies built in the container can run on any compute environment, including desktops, physical servers, virtual servers, testing, staging, production environments and public or private clouds. This portability makes hybrid and moving between cloud platforms more seamless.

ISOLATION

Containers do not interact with each other. If one microservice crashes, other containers in the same application will not be impacted. This feature also comes in handy in the case of malware or nefarious hacking. The hack will be isolated to that container and not take down the greater application. Hacked container is automatically replaced by a clean container.

LIGHTWEIGHT

Without the weight of the OS, containers can start up in seconds compared to minutes for most VMs. By taking up much less space, a higher utilization can be achieved. Developers can cram much more into a smaller area to lower either bare metal, data center or cloud costs.

NEED FOR SPEED

For enterprise wanting to innovate quickly and practice continual improvements to their applications containers are ideal. Customer experience is enhanced as bugs are fixed or new features are added on the fly without disruption to greater ecosystem. Within seconds containers can be created, replicated or destroyed speeding up the development process and time to market.

HORIZONTAL SCALING

With the correct design, containers can handle increased demand by adding identical containers within a cluster to scale out. This feature offers cost savings as just the containers needed in real time will scale, leaving the rest of the infrastructure available for other services.

DEVELOPER PRODUCTIVITY

Since containers run the same on a desktop as they do in the cloud, developers can achieve efficiencies when it comes to application design and testing. Not having to address environmental inconsistencies allows the developer to focus microservice health and spend less time debugging across environments.

With containers, development teams have been reorganized. Instead of owning a functional component of the application, they now own a microservice from end-to-end. The delegation of ownership and accountability contributes to strong application design with in turn enhances uptime. To update an application, developers simply iterate on their application code, create new containers, commit those containers to their container repository, often triggering an automated deploy. Version control allows for rolling out and rolling back in real time. Plus sharing containers using AWS's Elastic Container Registry (ECR) is simple, automatically giving their project team access to a developers work.

DO ONE THING AND DO IT WELL

Employing DevOps philosophies, the developer teams behind each of the bounded, yet independent microservices, own their microservice from build, to test, to release.

For example if an ecommerce container needs an upgrade or patch, that can be accomplished without disruption to the other containers within the application. With containers, one development team cannot break the code of the entire app. If code is broken it would be isolated to their own microservice. And with versioning, a container roll back can minimize any pain or user experience disruption.

The size and complexity of a cloud environment impacts the amount of containers needed. Netflix is AWS's biggest customer, and has over 700 microservices allowing for responsive, secure, architecture.

In the abstract containers are about deploying exactly the cloud resources you will need to complete a specific compute objective. For applications running in the cloud, microservices running in containers have enabled next level agility, flexibility and security. DevOps shops maximizing container technology also find that cloud infrastructure resources are used more

CONTAINER ANATOMY & CHARACTERISTICS

API Gateway

By interacting with the UI (User Interface) end users make requests to the API gateway, which reroutes requests to the correct microservice within a container. Containers within an application communicate with each other via APIs.

Owned by DevOps Team

With a product mindset (and eschewing the project mindset) DevOps teams build better applications. By marrying the technical product and business capabilities, stronger software emerges throughout the build, test and run states.

Smart End Points/ Dumb Pipe

By moving business logic into endpoints and using simple communication means like HTTPS, organizations can simplify infrastructure and jettison enterprise service bus architecture.

Decentralized Governance

Developers are in the driver's seat with their specific containerized microservices. Whether it is the programming language, the deployment methodology, the UI, teams are unburdened of legacy architecture or mindset. With pride in ownership, DevOps team build the best service possible.

Decentralized Data Management

With monolithic application design, the entire application will use one central database. With containers each microservice has their own data persistence. Inaccessible to other microservices, the private data of each microservice can only achieve read or write capability through a related component interface. With database isolated at the microservice layer, the amount of data available at a breach is factored down.

Infrastructure Automation

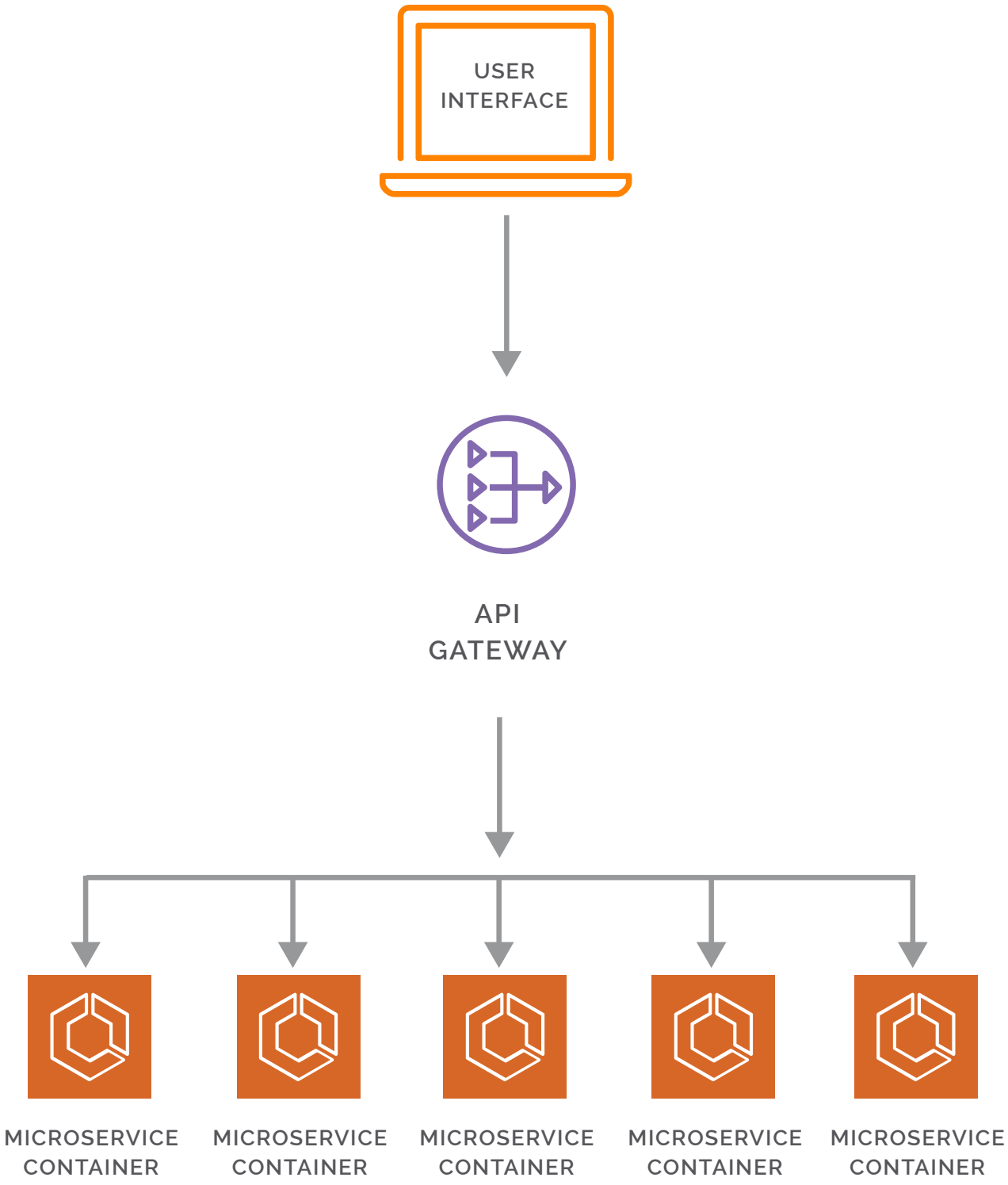
With a zero downtime methodology, blue/ green deployments and automation, containerization allows for the ability to spin up services with the press of a button. Monitoring tools discover issues quickly. Automated tools enable the replacement of containers and the auto healing of the service.

Fault Tolerance

With containers, and distributed architecture, issues are isolated and prevented from cascading throughout the ecosystem. The fault tolerance for each microservice can be defined in its manifest, ensuring that the service can survive underlying infrastructure component failures. This is a boon for uptime and security.

Designing for Failure

Container are cattle and not pets. Developers have an emotional detachment from the product and if one fails it is easily discarded and replaced by healthier version. The fact that containers are not interactive forces developers to capture all application requirements in code, like dockerfiles and K8S manifests, ensuring a 100% configuration as code environment with virtually zero drift.


USER
INTERFACE
API
GATEWAY
MICROSERVICE
CONTAINER
MICROSERVICE
CONTAINER
MICROSERVICE
CONTAINER
MICROSERVICE
CONTAINER
MICROSERVICE
CONTAINER

CONTAINER SECURITY

As with any architecture security around containers attracts much scrutiny. As container technology has just recently hit the tipping point for ubiquitous adoption, the inhouse engineering skills gap can impact security surety, and therefore slow adoption and innovation. With AWS Security and DevOps Competency partner like Foghorn, a wealth of container security expertise can deliver secure and compliant architecture that also delivers upon the promise of DevOps.

While containers present some more complexity from a security surface area point of view, they can also enhance security. By moving to smaller microservices within containerized applications compared to monolithic applications, even more fine grained IAM is available. If a vulnerability is exploited, it is isolated to the image and just the limited capabilities of that function. On identification, the corrupted image can be immediately replaced with a clean image. The attacker is stymied, as the application regains full functionality quickly.

BUILDING SECURE CONTAINER CHECKLIST

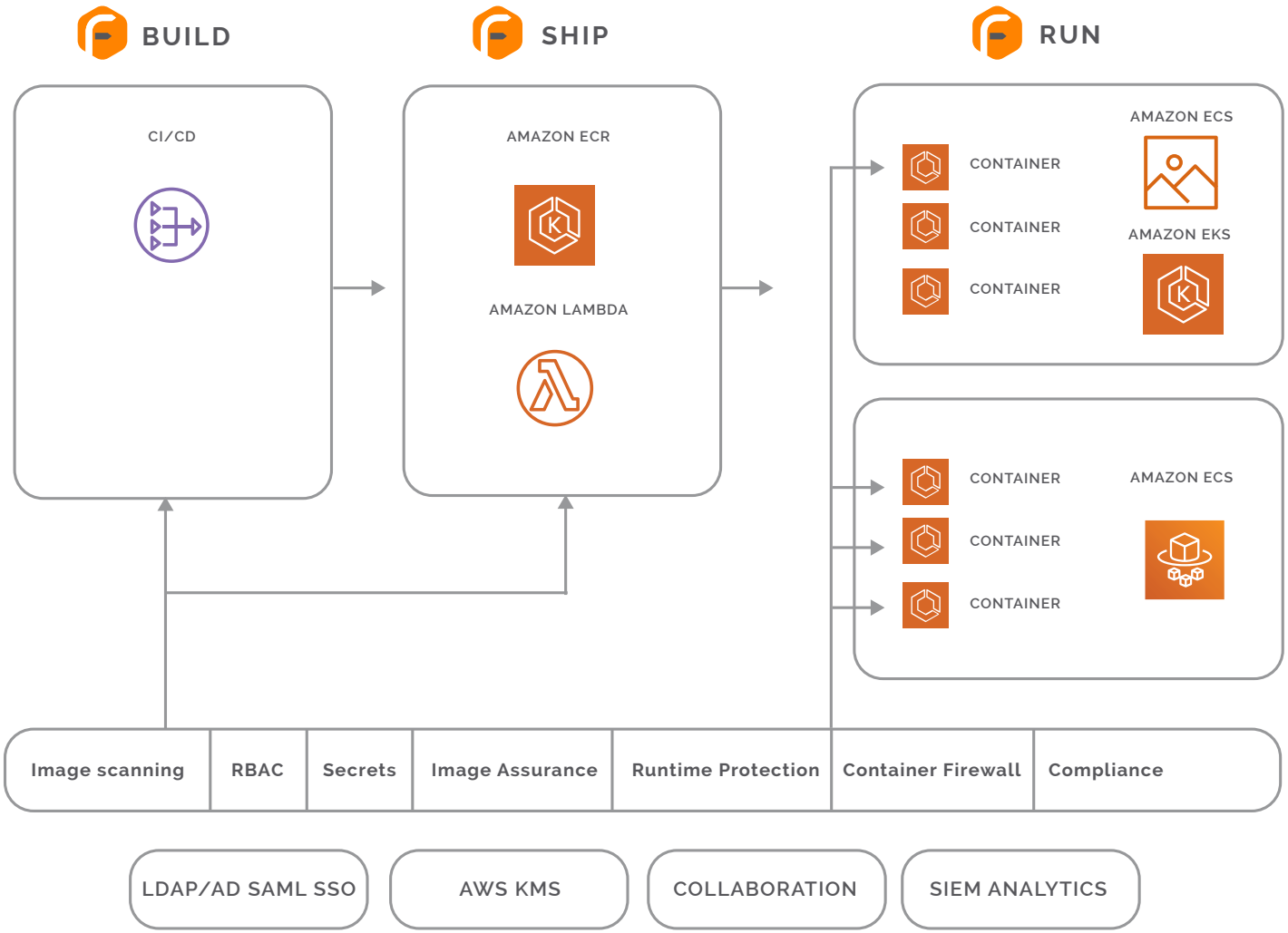
- ☐ Establish a container build pipeline.
- ☐ Incorporate security scanning into build pipeline.
- ☐ Offer hardened, approved, 'base images' to developers to ensure that development occurs with security controls in place.
- ☐ Establish deployment pipelines to test, staging, and production environments that allow DevOps teams to supply runtime requirements via manifests.

Compared to VM's, distributed containers (and the microservice within), can make network security more complex. With increased API calls from random ports across many servers, traditional layer 3 security devices and techniques are often inadequate for the job.

Application service mesh technologies like AWS app mesh and Istio for Kubernetes can efficiently manage transparent authorization and encryption, and agent based intrusion detection systems can bring monitoring and alerting to ephemeral networks.

AWS NATIVE TOOLS FOR CONTAINERS

As DevOps has matured, images running on containers have also matured, ensuring security and compliance. With the depth and breadth of options available within the AWS ecosystems to build, ship and run containers let's dive deeper into the exciting tools available on AWS.



Amazon Elastic Container Registry (ECR) is a Docker container registry designed to store, encrypt, and manage container images enabling quick start up time and global availability. IAM resource based policies ensure compliance. Images are transferred via SSL, and are encrypted at rest. ECR is backed by S3 with tight integration into ECS.

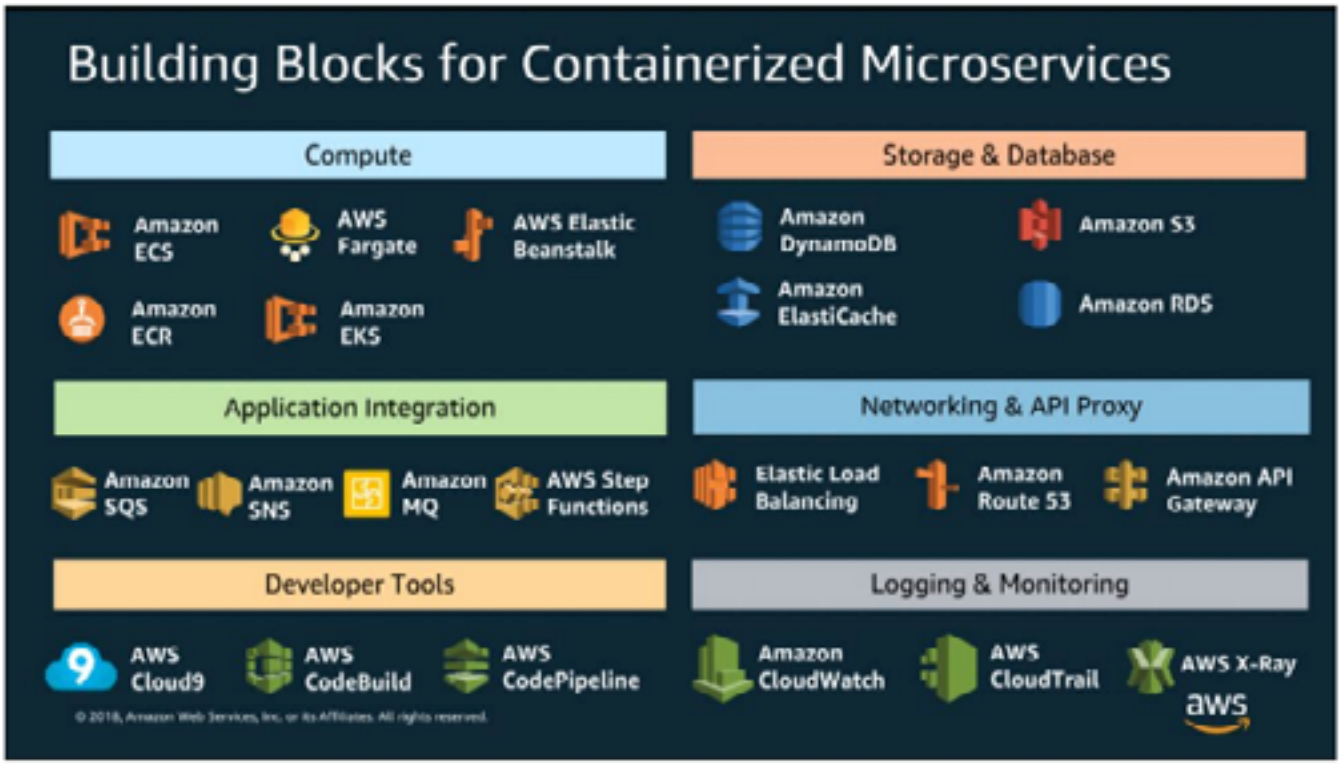
Amazon Elastic Container Service (ECS) is the AWS native container orchestration solution to run containerized applications or build microservices. There is no need to install and operate your own container orchestration software.

Amazon Elastic Container Service for Kubernetes (EKS) taps into the power of open sourced container orchestration tool Kubernetes. EKS integrates your Kubernetes control panel into the AWS ecosystem, enabling scalable containerized applications with speed, agility and security at their core.

Amazon Fargate scales and manages the servers required to run your containers. Scale your clusters, or optimize cluster packing with automated infrastructure provisioning enabling developers to focus even more keenly on microservice design and security. For containers AWS Fargate is a strong solution, while serverless solution Amazon Lambda excels for serverless execution for simple functions and the compute layer.

Amazon CodeDeploy is an excellent tool to launch a new version of your containerized application. With this blue/ green deployments, a new version can be put into a production test alongside the older version. Once the tests come back positive traffic can be rerouted to new version. If there is an issue your can quickly rollback to previous version.

AWS has built best in breed solutions to maximize the value of a container investment for your enterprise. From compute to storage to logging and monitoring finding the right mix is essential for ongoing success on AWS.



CONCLUSION

It may seem daunting to transition from monolithic applications to containerized applications made up of microservices. For many applications containers may not be the best fit. For those applications that do work well with containerization, unmatched flexibility and scalability are possible. By utilizing container best practices CISO's and regulators realize that containers provide a secure and compliant way to deliver scalable, distributed and high availability applications.

KEY BENEFITS:

> Containers make it easier to scale, maintain and evolve an application.

> Build/ test/ release cycles can be sped up.

> With clear ownership and accountability, teams can quickly iterate to add new features.

> Cloud infrastructure is utilized more effectively and efficiently.

Containers separate the application's baby from the bathwater, encouraging security and efficiency in build, test and run. DevOps teams can divide and conquer and eventually create an ecosystem that is easier to provision, deploy and heal. The increased agility breeds more effective infrastructure utilization, a higher velocity of innovation, and more delighted customers.

FOGHORN DELIVERS BUSINESS TRANSFORMATIONS ON AWS

Whether you are new to AWS or have an existing AWS environment you are looking to optimize, Foghorn can help. For over 10 years we have delivered outstanding results for clients on AWS. From DevOps in the Cloud to Security in theCloud, Foghorn has the talent, experience and credentials to deliver a velocityof innovation designed to increase performance while optimizing costs

aws partner network

Premier
Consulting Partner

DevOps Competency
Security Competency
Solution Provider

AWS DEVOPS COMPETENCY

When the silos are knocked down and development and operations can collaborate in a cycle of continuous improvement and continuous development in the cloud amazing things happen. Foghorn has been at the forefront of the DevOps revolution and are proud to have earned AWS DevOps Competency. When DevOps meets Foghorn, you have FogOps® and the promise of the cloud- Delivered.

AWS SECURITY COMPETENCY

Foghorn knows cloud security and DevSecOps in the cloud. In 2017 AWS launched a security competency to highlight their partners who satisfy and exceed AWS Cloud security best practices. The framework for this certification covers incident response, logging and monitoring, security, access management and data protection. Foghorn delivers DevSecOps results for customers from HIPPA/HITECH to PCI.

DEVELOP A CONTAINER STRATEGY TODAY

SCHEDULE A CALL

Foghorn Consulting was founded in 2008 with a mission to ensure that cloud computing initiatives deliver maximum value for its customers. Based in the Silicon Valley, Foghorn provides domain expertise in strategy, planning, execution and managed cloud services to high-growth and enterprise companies seeking a cloud partner. Our team of DevOps engineers, SRE's and certified cloud architects bring over 20 years of domain expertise to ensure your cloud initiatives are a success.

FOGHORN

330 Townsend St, Suite 202

San Francisco, CA 94107

foghornconsulting.com

info@foghornconsulting.com

650-963-0980