sysdig

Securing Containers on AWS Fargate Checklist

AWS Fargate is a popular serverless compute engine for containers that works with both Amazon ECS and Amazon EKS. Fargate makes it easy for you to focus on building your applications. It removes the need to provision and manage servers, and lets you specify and pay for resources per application.

While serverless environments free you to focus on modern application development there are also some challenges to be addressed. Serverless environments introduce an abstraction layer that hides the underlying infrastructure from the DevOps and security teams. Without access to the host or traditional monitoring tools, your visibility into workload activity can be limited, leaving you blind to threats. Once you identify a security or performance issue, teams need a detailed record of activity to respond to incidents and troubleshoot issues.

No visibility into the underlying infrastructure

Can’t alert on vulnerabilities and threats

Need a detailed record of exactly what happened after the Fargate task is gone

The key is to continuously scan for cloud and container vulnerabilities, detect abnormal activity, reduce your risk from cloud misconfigurations and prioritize threats to ensure your containers on Fargate are secure across their entire life cycle. These five key workflows will enable you to address the most critical security and visibility requirements so you can confidently and securely run containers on AWS Fargate.

1

SECURING CONTAINERS ON AWS FARGATE CHECKLIST

Automatically scan Fargate containers in your cloud

As the number of container images, versions, and builds proliferates, you lose control of what software is being used and whether software updates are applied. Embedding security into your delivery pipeline as you build applications helps you identify and address vulnerabilities faster, and keeps your developers productive. You can start by automatically triggering scanning within your Elastic Container Registry (ECR), and then take these steps to maintain continuous control:

- Automatically scan images tied to a Fargate task and prevent risky images from being deployed.
- Protect sensitive data by directly scanning images in your cloud account.
- Validate the build configuration (Dockerfile instructions) and image attributes (like size and labels).
- Identify new vulnerabilities that impact the image once the container has been deployed.
- Create different policies for each workflow, including images from public repositories and images built in-house. Consider different checks for each app.
- Alert the right team for each issue (notify the owner of each image and integrate with your CI/CD tool to show the scan results directly in that context).

By integrating security analysis and compliance validation into this process, you can address issues earlier so you don't slow down deployment. This is known as "shifting security left."

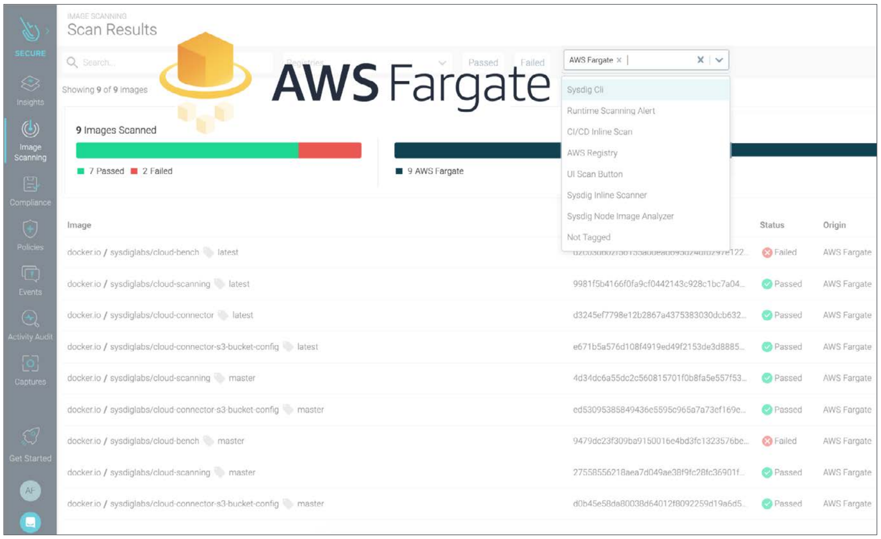


IMAGE SCANNING
Scan Results

Showing 9 of 9 images

9 Images Scanned

7 Passed 2 Failed

9 AWS Fargate

Image		Status	Origin
docker.io / sysdiglabs/cloud-bench latest	[illegible]	Failed	AWS Fargate
docker.io / sysdiglabs/cloud-scanning latest	9981f5b4166f0fa9cf0442143c928c1bc7a04...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-connector latest	d3245ef7798e12b2867a4375383030dcb632...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-connector-s3-bucket-config latest	e671b5a576d108f4919ed49f2153de3d8885...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-scanning master	4d34dc6a55dc2c560815701f0b8fa5e557f53...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-connector-s3-bucket-config master	ed53095385849436e5595c965a7a73cf169e...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-bench master	9479dc23f309ba9150016e4bd3fc1323576be...	Failed	AWS Fargate
docker.io / sysdiglabs/cloud-scanning master	27558556218aea7d049ae38f9fc28fc36901f...	Passed	AWS Fargate
docker.io / sysdiglabs/cloud-connector-s3-bucket-config master	d0b45e58da80038d64012f8092259d19a6d5...	Passed	AWS Fargate

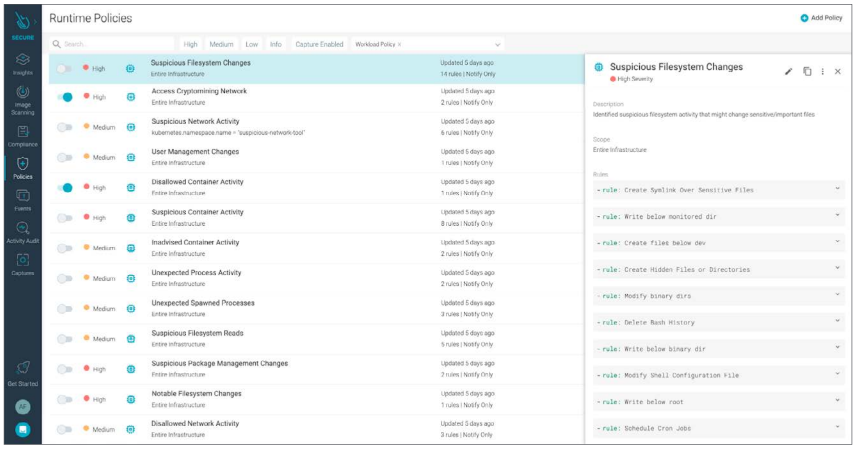
2

SECURING CONTAINERS ON AWS FARGATE CHECKLIST

Detect and respond to runtime threats

You can reduce runtime risk by configuring applications with minimum privilege and access permissions. Your policies should also monitor for anomalous behavior and configuration drift. Creating policies that can prevent attacks without breaking the applications is challenging. Be sure to capture a detailed record for incident response even after the Fargate task is gone. Consider these steps for reducing your runtime risk.

- Create and maintain runtime security policies that observe workload behavior, monitor cloud activity, and identify anomalous events in Fargate.
  - Leverage tools to automatically build and customize policies or use out-of-the-box Falco rules.
  - Implement least-privilege and compliant network policies with Kubernetes and app metadata.
  - Visualize network communication in and out of a particular pod/service/app/tag over time with topology maps.
  - Automate use of events in AWS CloudTrail logs to detect threats and configuration changes on cloud services.
- Monitor Fargate task usage to detect anomalous activity. Tracking network connections gives you information about the attack, runtime behavior, and spread vectors. Some attacks are first detected as monitoring alerts rather than security violations.
- Streamline incident response and quickly respond to container and cloud security threats with a detailed activity record in Fargate. Make sure you have a unified view of cloud and container threats across your multi cloud environment. Use capture files based on syscall data to quickly answer the questions of “when,” “what,” “who,” and “why” for your container security incidents. This detailed record allows you to conduct post-mortem analysis and determine root cause, even after containers are gone.



Runtime Policies

High	Suspicious Filesystem Changes Entire Infrastructure	Updated 5 days ago 14 rules | Notify Only
High	Access Cryptomining Network Entire Infrastructure	Updated 5 days ago 2 rules | Notify Only
Medium	Suspicious Network Activity kubernetes.namespace.name = 'suspicious-network-tool'	Updated 5 days ago 6 rules | Notify Only
Medium	User Management Changes Entire Infrastructure	Updated 5 days ago 1 rules | Notify Only
High	Disallowed Container Activity Entire Infrastructure	Updated 5 days ago 1 rules | Notify Only
High	Suspicious Container Activity Entire Infrastructure	Updated 5 days ago 8 rules | Notify Only
Medium	Inadvised Container Activity Entire Infrastructure	Updated 5 days ago 2 rules | Notify Only
Medium	Unexpected Process Activity Entire Infrastructure	Updated 5 days ago 2 rules | Notify Only
Medium	Unexpected Spawned Processes Entire Infrastructure	Updated 5 days ago 3 rules | Notify Only
Medium	Suspicious Filesystem Reads Entire Infrastructure	Updated 5 days ago 5 rules | Notify Only
High	Suspicious Package Management Changes Entire Infrastructure	Updated 5 days ago 2 rules | Notify Only
High	Notable Filesystem Changes Entire Infrastructure	Updated 5 days ago 1 rules | Notify Only
Medium	Disallowed Network Activity Entire Infrastructure	Updated 5 days ago 3 rules | Notify Only

Suspicious Filesystem Changes
High Severity

Description
Identified suspicious filesystem activity that might change sensitive/important files

Scope
Entire Infrastructure

Rules

```
- rule: Create Symlink Over Sensitive Files
- rule: Write below monitored dir
- rule: Create files below dev
- rule: Create Hidden Files or Directories
- rule: Modify binary dirs
- rule: Delete Bash History
- rule: Write below binary dir
- rule: Modify Shell Configuration File
- rule: Write below root
- rule: Schedule Cron Jobs
```

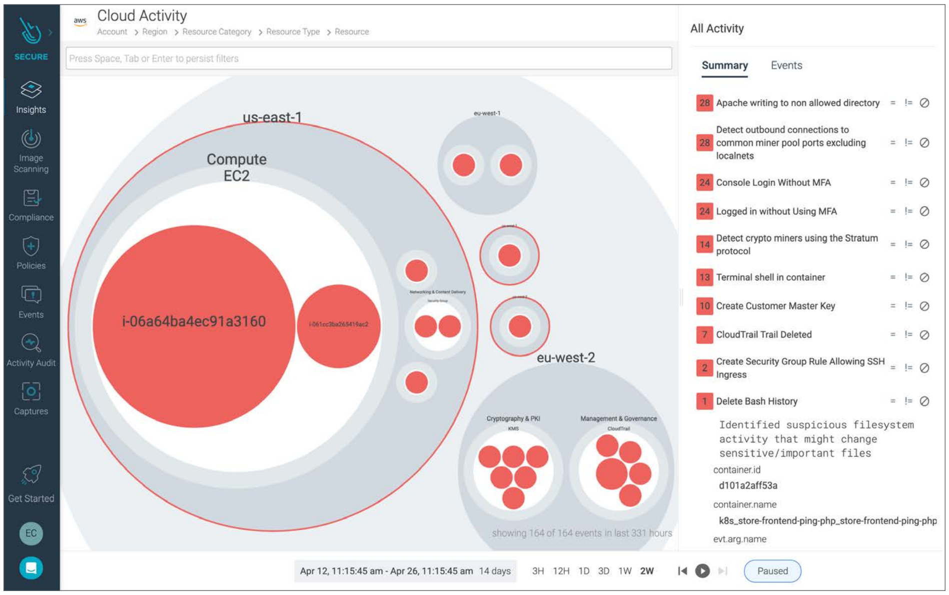
3

SECURING CONTAINERS ON AWS FARGATE CHECKLIST

Continuously manage Fargate posture

Continuous cloud security is required to immediately identify configuration errors and suspicious behavior. According to the shared responsibility model it is the responsibility of AWS users to secure their application and data. The following steps can help you validate the security posture for your Fargate workloads:

- Automatically discover and inventory the AWS assets running in your environment including systems, applications, and services like VPCs, RDS, S3 buckets, ECS, EKS, and Fargate.
- Flag misconfigurations based on open-source Cloud Custodian rules. Check your cloud configuration periodically against Center for Internet Security (CIS) benchmarks to identify risky configuration settings (e.g., public storage buckets, exposed security groups and access controls, etc.) and take steps to remediate violations.
- Detect unexpected changes and suspicious activity e.g changing access rights to sensitive data across all cloud accounts, users, and services by parsing AWS CloudTrail logs using open-source Falco rules. Investigate all suspicious activity performed by a specific user to see the breadth of impact.



28	Apache writing to non allowed directory	= != ⊘
28	Detect outbound connections to common miner pool ports excluding localnets	= != ⊘
24	Console Login Without MFA	= != ⊘
24	Logged in without Using MFA	= != ⊘
14	Detect crypto miners using the Stratum protocol	= != ⊘
13	Terminal shell in container	= != ⊘
10	Create Customer Master Key	= != ⊘
7	CloudTrail Trail Deleted	= != ⊘
2	Create Security Group Rule Allowing SSH Ingress	= != ⊘
1	Delete Bash History	= != ⊘

```
Identified suspicious filesystem
activity that might change
sensitive/important files
```

container.id
d101a2aff53a

container.name
k8s_store-frontend-ping-php_store-frontend-ping-php

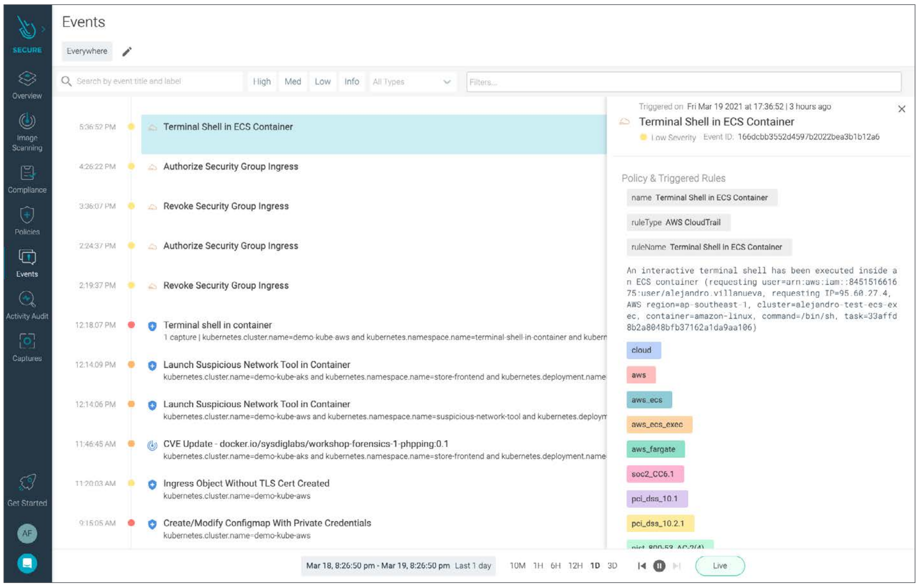
evt.arg.name

4

Continuously validate compliance

Implement compliance checks to meet regulatory compliance standards (CIS, SOC2, PCI, NIST 800-53, etc.) across containers, Kubernetes, and cloud and benchmark against best practices. Monitor cloud services continually for configuration drift that can impact compliance. Measure compliance progress with scheduled assessments and detailed reports.

- Check your container and platform configuration against CIS benchmarks for AWS, Docker, and Kubernetes.
- Validate compliance during build-time by mapping container image scanning policies to standards (e.g., NIST, PCI, SOC2, or HIPAA) or internal compliance policies (e.g., blacklisted images, packages, or licenses).
- Implement File integrity Monitoring (FIM) to detect tampering of critical system files, directories, and unauthorized changes. FIM is a core regulatory requirement for a number of compliance standards.
- Manage compliance at runtime as well. Check for best practices (e.g., don't run privileged containers and don't run containers as root) and look for known adversary tactics and techniques. Achieve and maintain compliance with security frameworks mapping through a rich set of Falco rules for security standards and benchmarks, like NIST 800-53, PCI DSS, SOC 2, MITRE ATT&CK®, CIS AWS, and AWS Foundational Security Best Practices.
- Provide proof of compliance with capture files that incorporate detailed forensics data. It's important to record configuration and policy changes, including an audit of runtime changes for compliance audits.



Events

5:36:52 PM Terminal Shell in ECS Container

4:26:22 PM Authorize Security Group Ingress

3:36:07 PM Revoke Security Group Ingress

2:24:37 PM Authorize Security Group Ingress

2:19:37 PM Revoke Security Group Ingress

12:18:07 PM Terminal shell in container
1 capture | kubernetes.cluster.name=demo-kube-aws and kubernetes.namespace.name=terminal-shell-in-container and kubern

12:14:09 PM Launch Suspicious Network Tool in Container
kubernetes.cluster.name=demo-kube-aks and kubernetes.namespace.name=store-frontend and kubernetes.deployment.name

12:14:06 PM Launch Suspicious Network Tool in Container
kubernetes.cluster.name=demo-kube-aws and kubernetes.namespace.name=suspicious-network-tool and kubernetes.deploym

11:46:45 AM CVE Update - docker.io/sysdiglabs/workshop-forensics-1-phpping:0.1
kubernetes.cluster.name=demo-kube-aks and kubernetes.namespace.name=store-frontend and kubernetes.deployment.name

11:20:03 AM Ingress Object Without TLS Cert Created
kubernetes.cluster.name=demo-kube-aws

9:15:05 AM Create/Modify Configmap With Private Credentials
kubernetes.cluster.name=demo-kube-aws

Mar 18, 8:26:50 pm - Mar 19, 8:26:50 pm Last 1 day

Triggered on Fri Mar 19 2021 at 17:36:52 | 3 hours ago

Terminal Shell in ECS Container

Low Severity Event ID: 166dcbb3552d4597b2022bea3b1b12a6

Policy & Triggered Rules

name Terminal Shell in ECS Container

ruleType AWS CloudTrail

ruleName Terminal Shell in ECS Container

```
An interactive terminal shell has been executed inside a
n ECS container (requesting user=arn:aws:iam::8451516616
75:user/alejandro.villanueva, requesting IP=95.60.27.4,
AWS region=ap-southeast-1, cluster=alejandro-test-ecs-ex
ec, container=amazon-linux, command=/bin/sh, task=33affd
8b2a8048bfb37162a1da9aa106)
```

cloud

aws

aws_ecs

aws_ecs_exec

aws_fargate

soc2_CC6.1

pci_dss_10.1

pci_dss_10.2.1

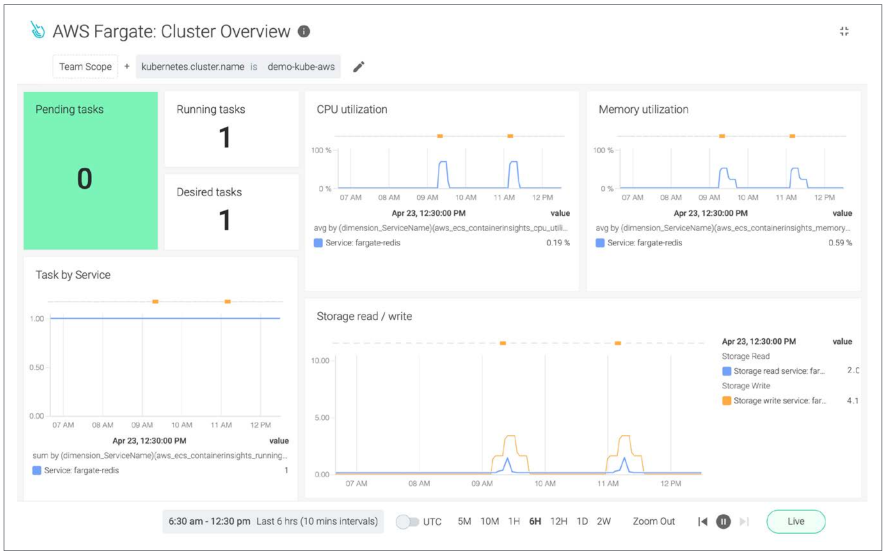
nist_800-53_AC-2(4)

5

Monitor and troubleshoot Fargate containers

Containers and cloud services are dynamic and churn constantly. Visibility into the health and performance of your AWS workloads and infrastructure is critical for ensuring the availability of your cloud applications.

- Implement monitoring built for cloud native infrastructure, applications, and AWS services. Microservices can be distributed across multiple instances, and containers can run across different regions and multi-cloud infrastructure. To improve application performance and rapidly solve issues, you need deep container, infrastructure, and service visibility and granular metrics enriched with Kubernetes and cloud context.
    - Immediately identify owners for issue resolution using container and cloud context.
    - Identify pods consuming excessive resources and monitor capacity limits.
- Tap into open-source Prometheus for monitoring AWS services and cloud-native applications. Extract Prometheus metrics via AWS CloudWatch for Fargate, and view with dashboards in Grafana.
    - Get productive quickly by using Promcat.io, a resource catalog of Prometheus integrations with curated, documented, and supported monitoring integrations for Kubernetes platform and cloud-native services.
- Capture and store data to ensure you can investigate and solve issues quickly. Once a container dies, everything inside is gone. You can't shell into a stopped container to see what happened! Audit logs and detailed activity information will help you successfully determine root cause, even for containers that are no longer running.



AWS Fargate: Cluster Overview

Team Scope + kubernetes.cluster.name is demo-kube-aws

Pending tasks 0

Running tasks 1

Desired tasks 1

CPU utilization
Apr 23, 12:30:00 PM value
avg by (dimension_ServiceName)(aws_ecs_containerinsights_cpu_utili...
Service: fargate-redis 0.19 %

Memory utilization
Apr 23, 12:30:00 PM value
avg by (dimension_ServiceName)(aws_ecs_containerinsights_memory...
Service: fargate-redis 0.59 %

Task by Service
Apr 23, 12:30:00 PM value
sum by (dimension_ServiceName)(aws_ecs_containerinsights_running...
Service: fargate-redis 1

Storage read / write
Apr 23, 12:30:00 PM value
Storage Read
Storage read service: far... 2.C
Storage Write
Storage write service: far... 4.1

6:30 am - 12:30 pm Last 6 hrs (10 mins intervals)

Dig deeper into how Sysdig can continuously secure your containers on AWS Fargate.

START YOUR FREE TRIAL

GET A PERSONALIZED DEMO

sysdig