CloudGuard Usecase Flow Diagrams

Table of Contents

Check Point
SOFTWARE TECHNOLOGIES LTD

CI/CD

Source code commit

Container image build

CloudGuard
Workload

ShiftLeft Image Scan

ShiftLeft IAC Scan

ShiftLeft Code Scan

Templates

GitHub Repos

CloudGuard
Workload

Container Protection

Serverless Function Protection

CloudGuard
Posture Management

Compliance & Governance

Cloudbots (Auto-remediation)

IAM Protection

CloudGuard
Intelligence

Log Enrichment

ML Abnormality Assessment

Public Cloud
Service Provider

aws Azure

Containers

Serverless

Cloud Roles
Users Permissions

Traffic Logs

Event Activity Logs

Applications

CloudGuard
Network Security

Advanced Threat Prevention

Automated Network Security

Multi-Cloud Management

< API >

Web traffic

CloudGuard
AppSec

Web Application Security

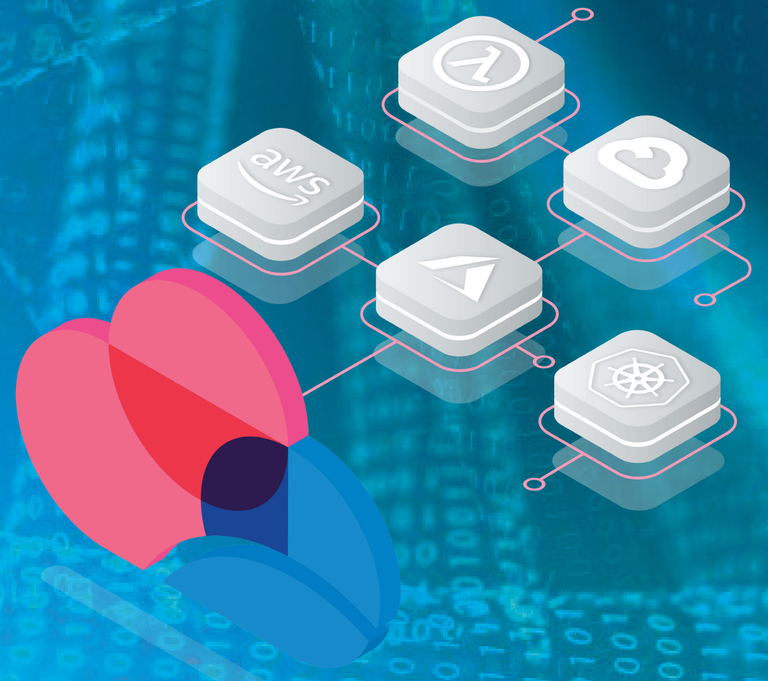
Secure API's

Bot Protection

VPN

Remote Users

DevSecOps



Source code commit

CI/CD

Continue pipeline

Input
Local Git Repository

CloudGuard
Workload

ShiftLeft Code Scan

Remediate issues found in source code

Package Manager

Vulnerability scanning
Check package managers for vulnerable dependencies

Code Analysis
Check for weak practices like SQL injection, cross side scripting, weak ciphers

File analysis
Detect sensitive content and exposed credentials

CLI or CG Portal
Summary of issues
Recommended Remediations

CP Backend
Parsed info like IPs (not files) sent to CP backend

IP/URL Reputation
Detect IPs and URLs hard coded in files for reputation

Analyze Malware
Using ThreatCloud API, analyze file reputation

License Restriction Scanning
Identify restricted dependency licenses

Code-scan Deep Dive:

Use Case: Ensuring Security in Source Code

ShiftLeft's code-scan blade is integrated into the CICD Pipeline.
When there is a source code commit, this blade receives a local Git repository as an input.
From there, many security scans are run.
If any issues are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues to the next stage.

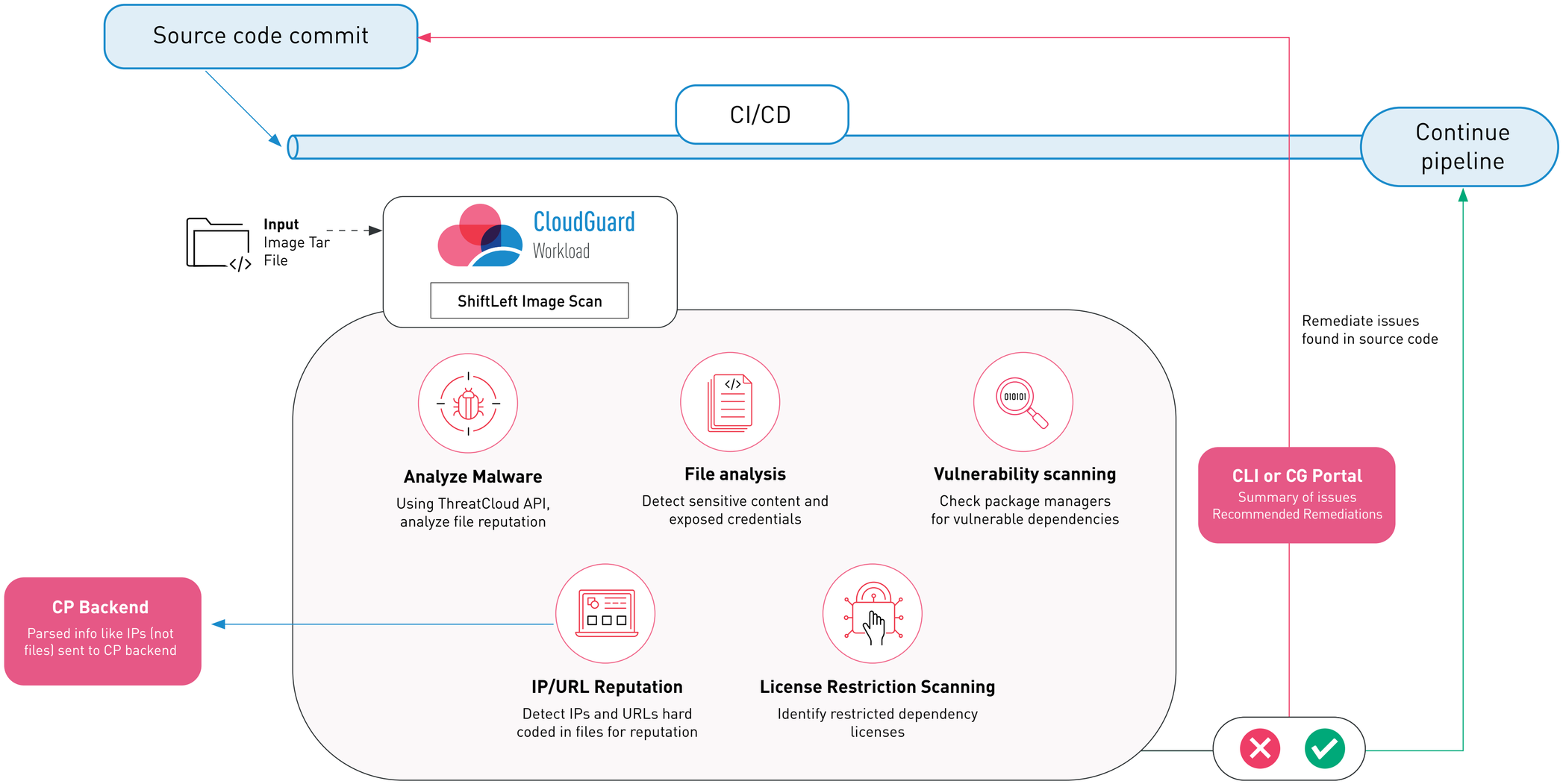
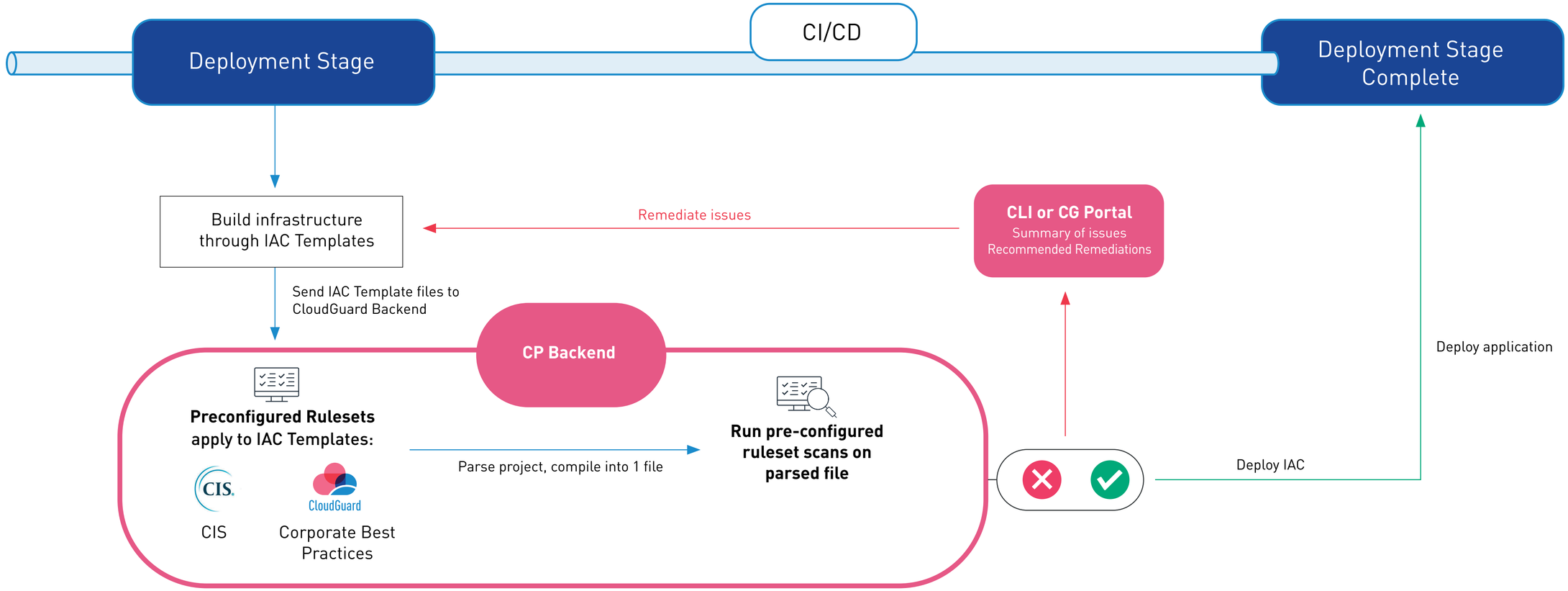


Image-scan Deep Dive:

Use Case: Ensuring Security in Container Images

ShiftLeft's image-scan blade is integrated into the CICD Pipeline.
When there is a container image build, this blade receives an image tar file as an input.
From there, many security scans are run.
One of these is vulnerability scanning, which checks OS level and application packages for vulnerabilities.
If a vulnerability is found, that information is sent to the CLI or CloudGuard Portal with a recommended package version to remediate.
If any other issues are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues to the next stage.



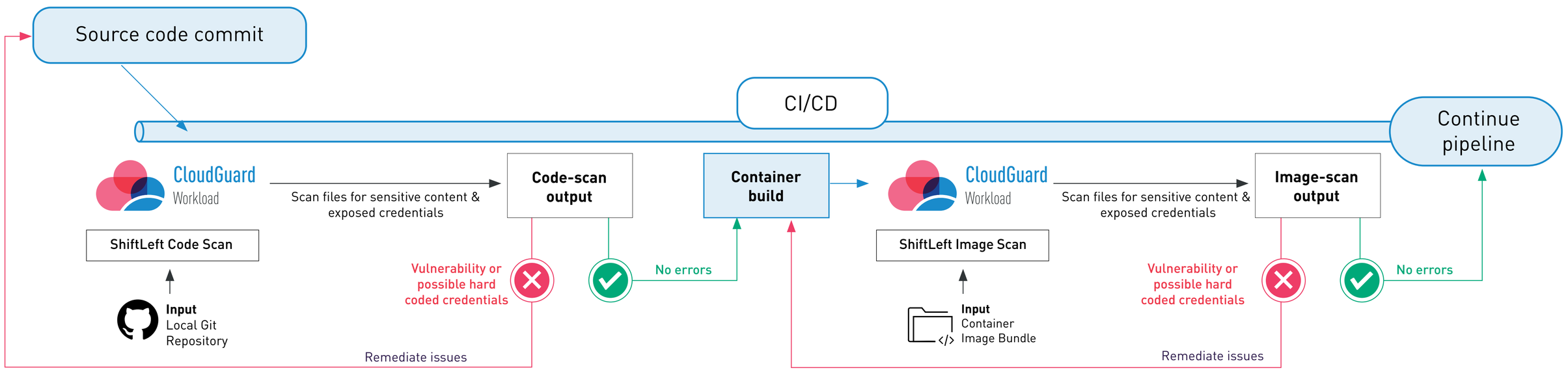
IAC-scan Deep Dive:

Use Case: Enforce Customizable Mandatory Policies to Infrastructure Deployments

ShiftLeft's IAC-scan blade can be used to enforce customizable mandatory policies to infrastructure deployments.
It is integrated into the deployment stage of the CICD pipeline, so that a build infrastructure event will send the IAC Template files to the CloudGuard backend.

Ahead of time and within the CloudGuard portal, users can configure rulesets to apply to these IAC Templates.
The CloudGuard backend parses the project and compiles it into one file, then runs these pre-configured ruleset scans on it.

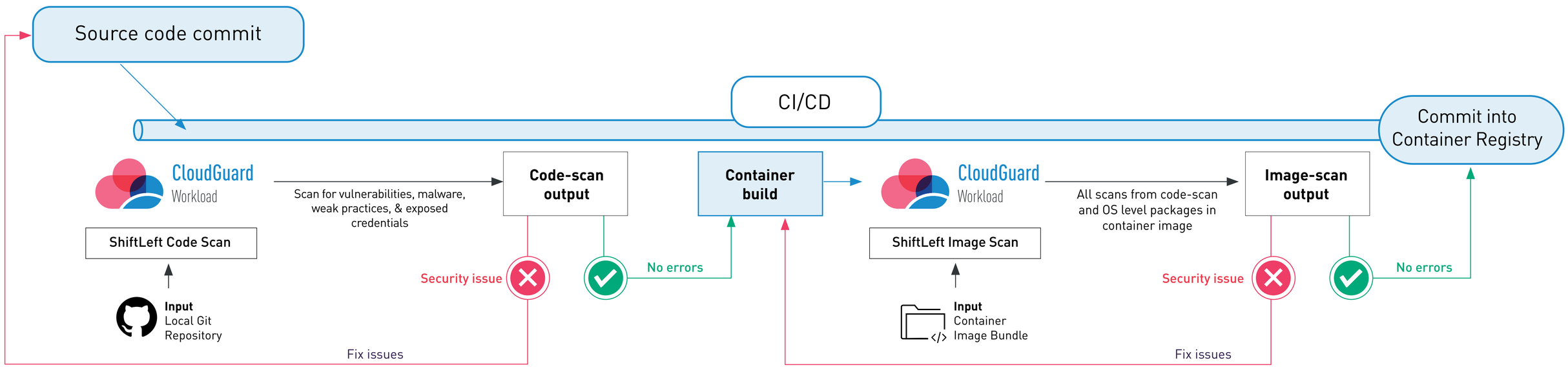
If any issues are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues to deploy the IAC and the application.



Credential Exposure:

Use Case: Uncovering Exposed Credentials in Code Repositories and Containers

CloudGuard's ShiftLeft tool can be used to check for exposed credentials and sensitive content.
It is integrated into the CICD Pipeline.
When there is a source code commit, the code-scan blade receives a local Git repository as an input.
From there, the files are scanned for sensitive content and exposed credentials.
If any possible hard coded credentials or vulnerabilities are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues to the next stage.
This process is repeated when there is a container build, except the image-scan blade receives a container image bundle as an input.

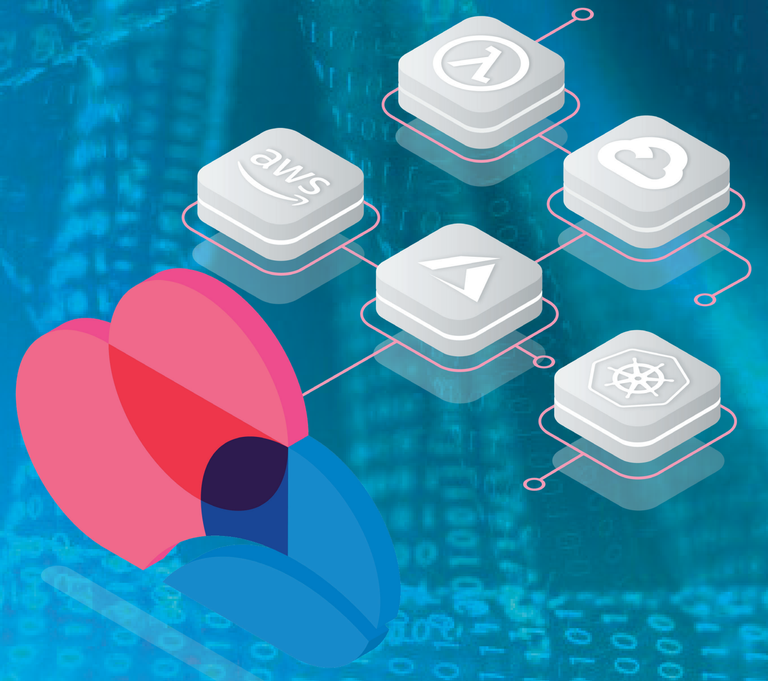


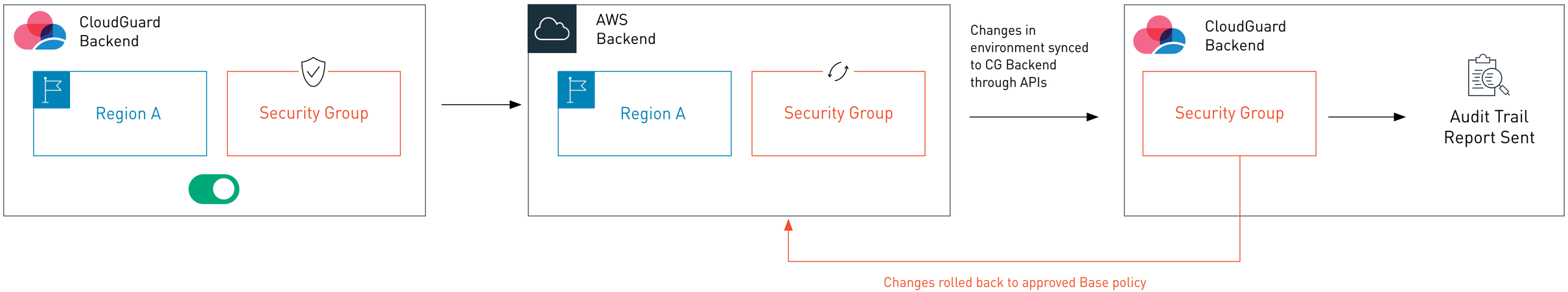
Container Security:

Use Case: Insert Container Security into pipeline in pre-production

The ShiftLeft tool is integrated into the CICD Pipeline and provides container security.
When there is a source code commit, the code-scan blade receives a local Git repository as an input.
From there, the files are scanned for vulnerabilities, malware, weak practices, and exposed credentials.
If any issues are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues to the next stage.
When there is a container build, the image-scan blade receives a container image bundle as an input.
It will run all the scans from the code-scan blade, as well as checking OS level packages in container images.
If any issues are found, the pipeline is stopped and an error message will appear showing what needs to be remediated.
If no issues are found, the pipeline continues with a commit into the container registry.

CSPM or Compliance/Governance



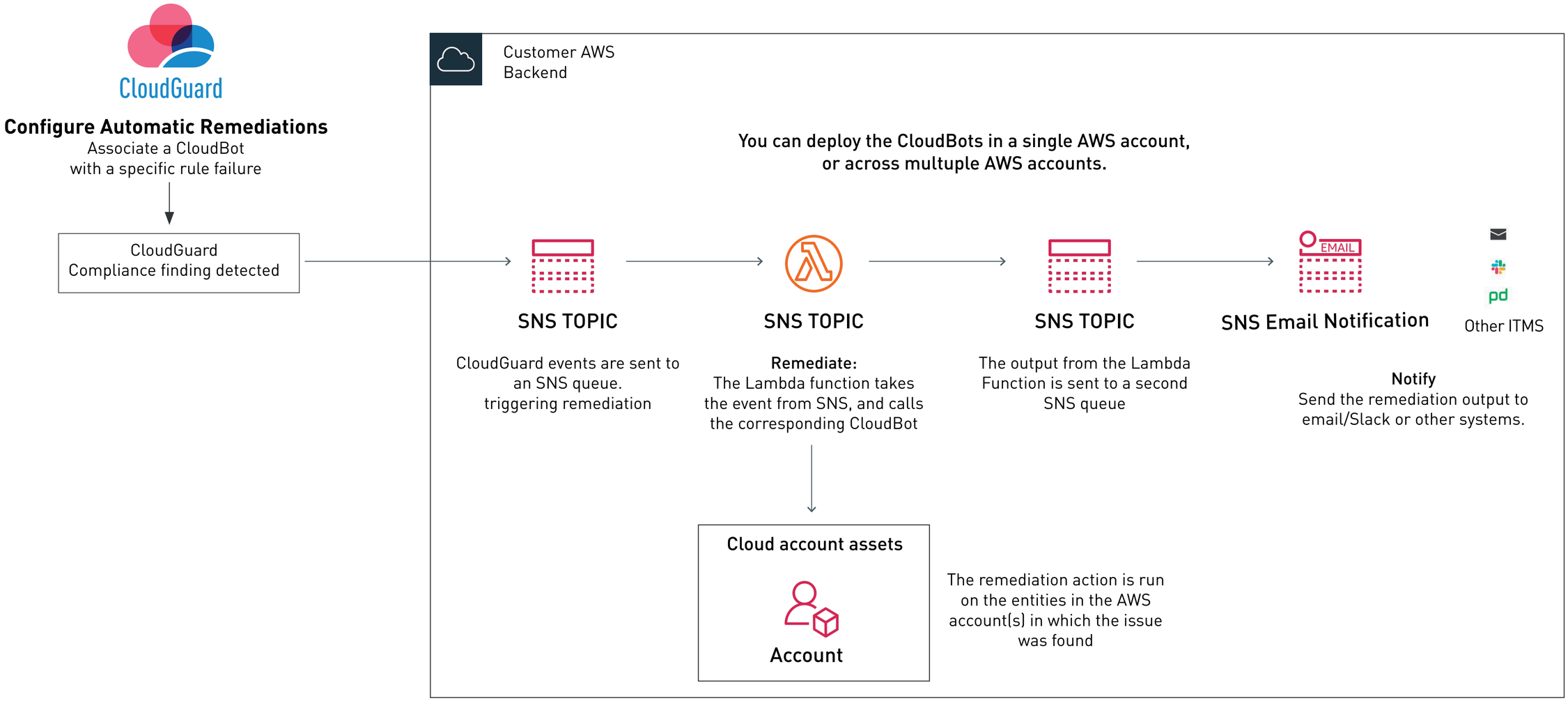


Cloud Security Posture Management:

Use Case: Tamper Protection for Security Groups in AWS

CloudGuard CSPM allows users to avoid changes to specific security groups in AWS. This prevents security groups from being made too permissive, thereby putting cloud accounts and the assets within them at risk.

1. When onboarding an AWS account to CloudGuard CSPM, users must set the account (or a specific region) to Full Protection mode. This will require giving CloudGuard some write permissions.
2. Once security admins have configured the security groups within that region to the base level they do not want altered, they can turn on the Tamper Protection feature for a specific security group or all security groups within a region
3. When a change is made to a security group which has tamper protection enabled, that information is synced to the CloudGuard backend through APIs.
4. At this point CloudGuard rolls back the changes to the approved base policy and will audit both the changes that were attempted and the rollback event.

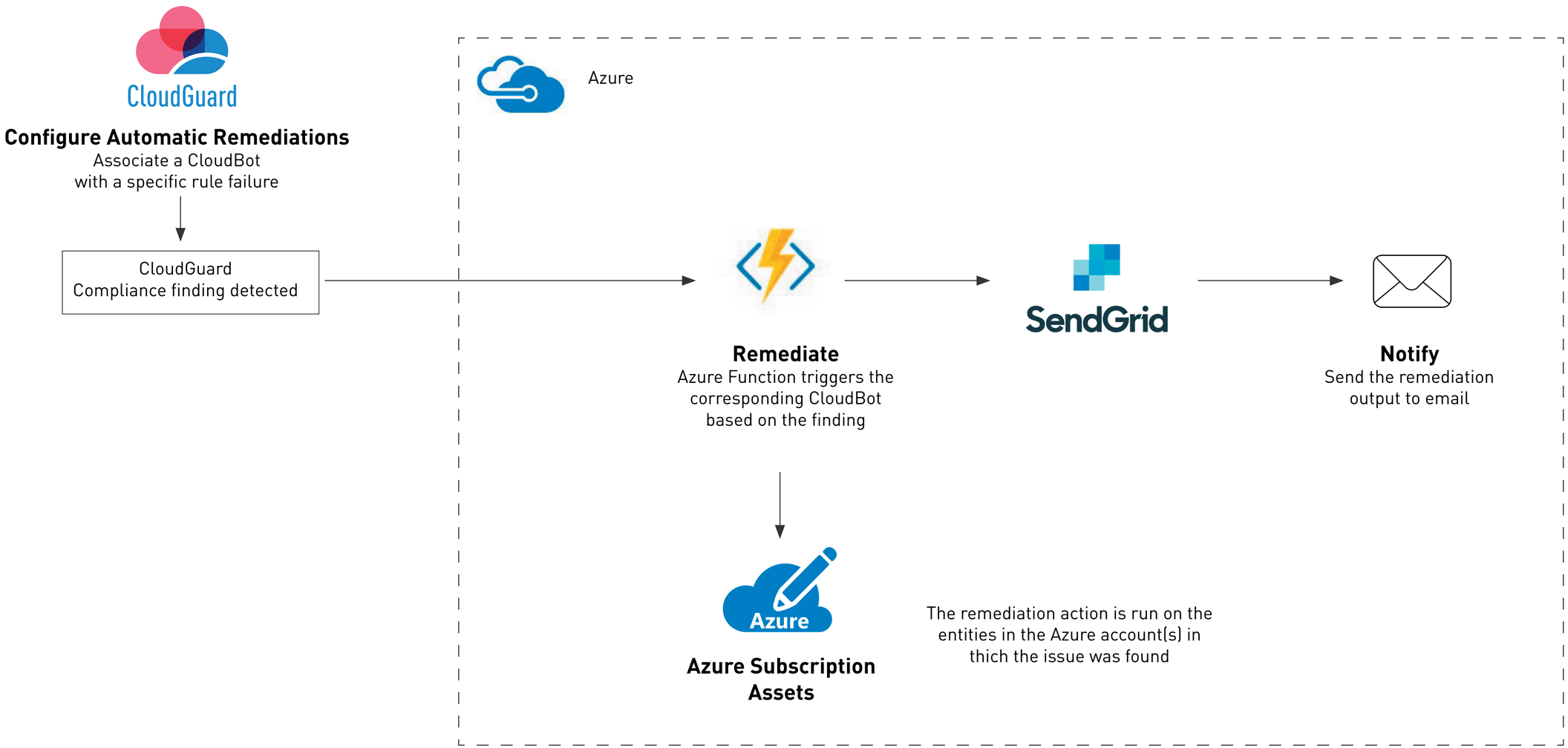


Compliance/Governance

Use Case: Auto-remediate failures from Compliance/Governance reports in AWS

CloudBots can be configured to automatically remediate any rule failures in CloudGuard rulesets for AWS. When a failure is detected, that event is sent to an SNS queue, triggering a remediation. The Lambda function takes the event from SNS, and calls the corresponding CloudBot.

The remediation action is run on the entities in the AWS account(s) in which the issue was found. For notification, the output from the Lambda Function is sent to a second SNS queue, and out to an SNS email notification. The remediation output is sent to email, Slack, or other systems.

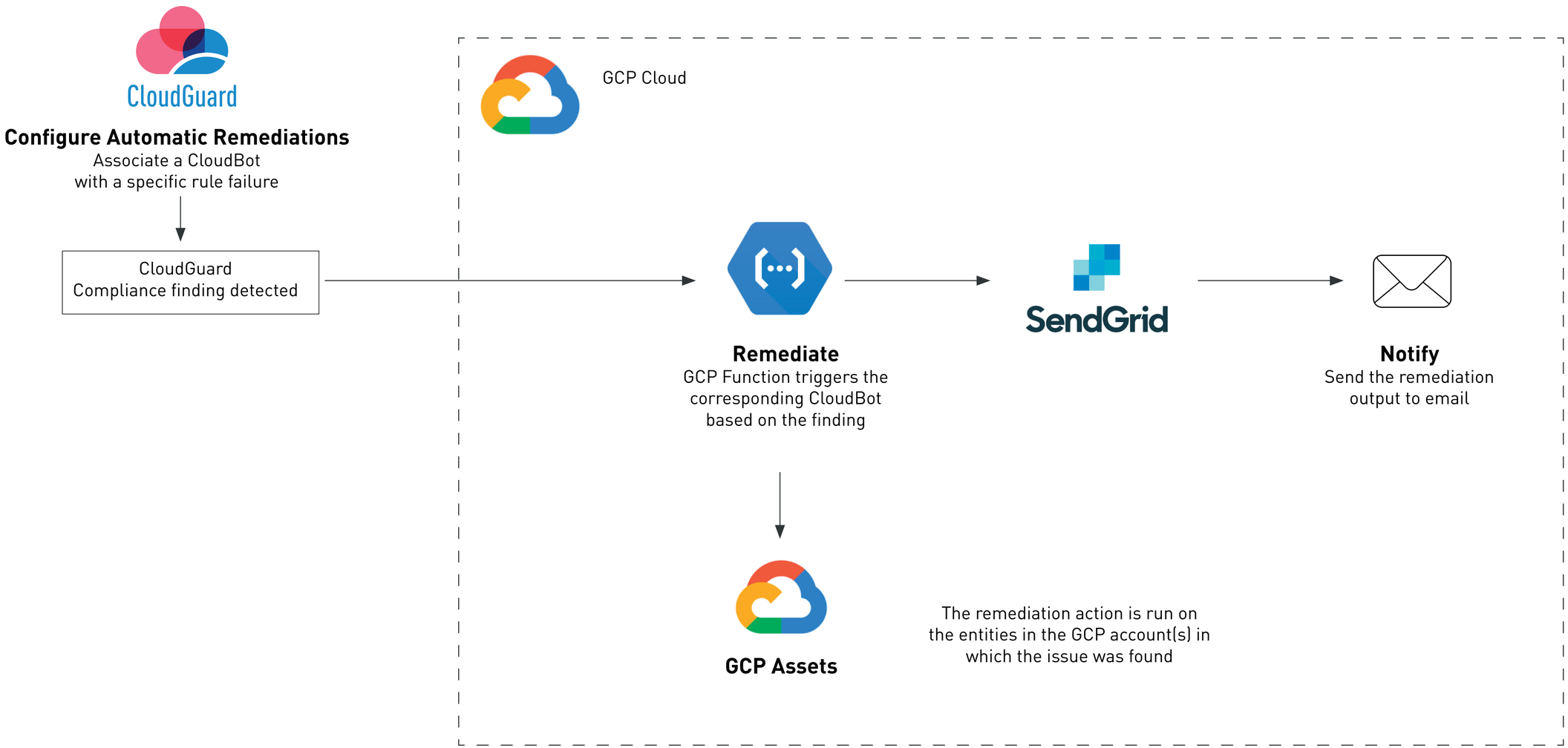


Compliance/Governance

Use Case: Auto-remediate failures from Compliance/Governance reports in Azure

CloudBots can be configured to automatically remediate any rule failures in CloudGuard rulesets for Azure.
When a failure is detected, an Azure function is triggered to launch the corresponding CloudBot based on the finding.

The remediation action is run on the entities in the Azure account(s) in which the issue was found.
SendGrid is used to send the remediation output to email.



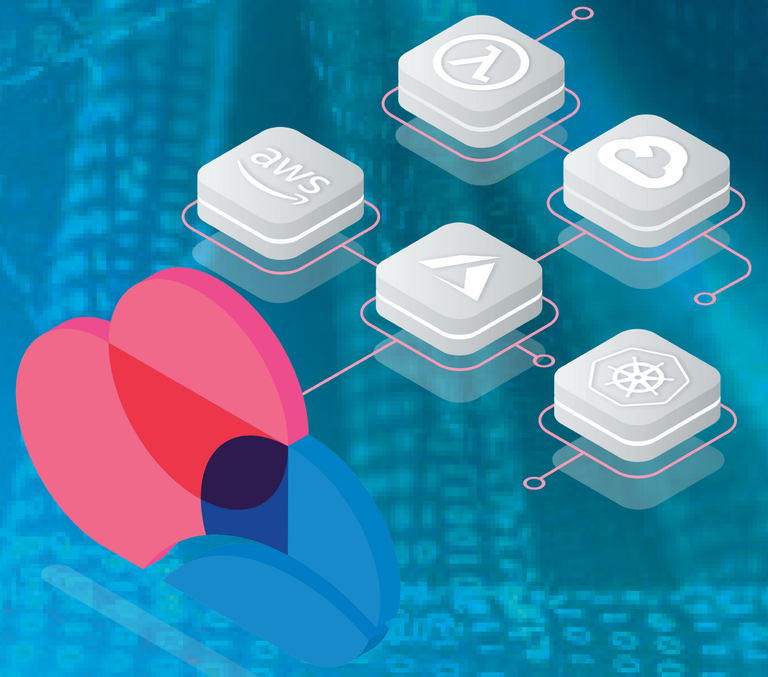
Compliance/Governance

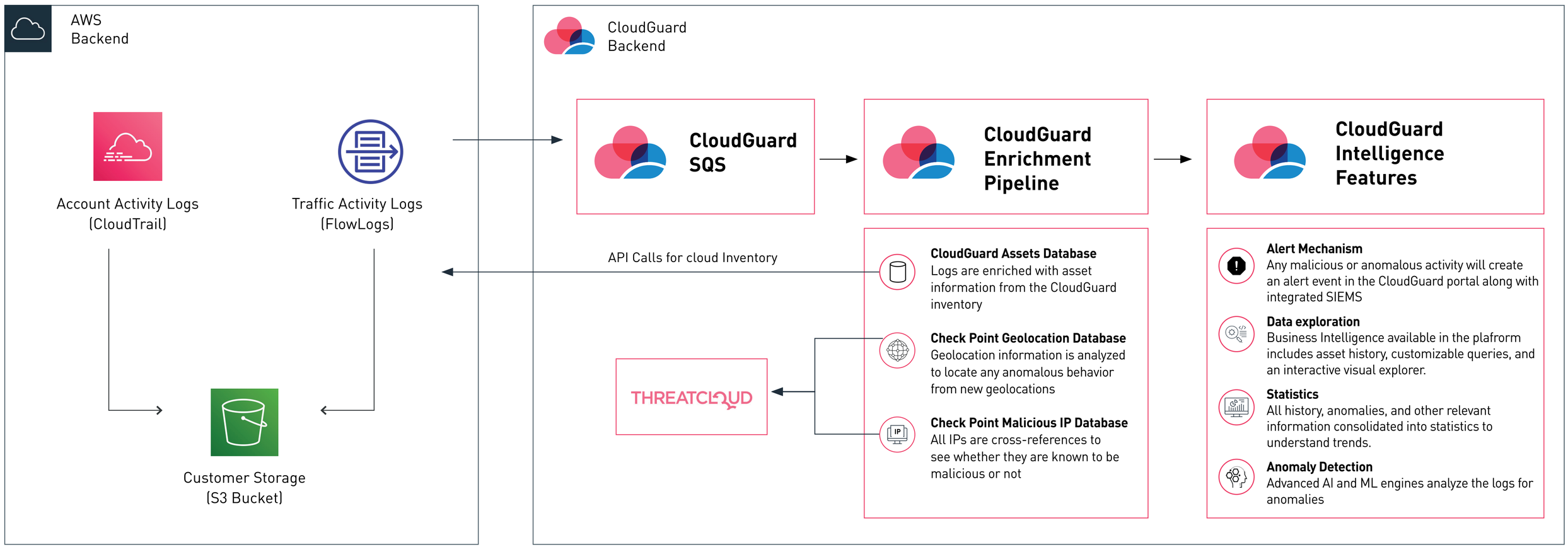
Use Case: Auto-remediate failures from Compliance/Governance reports in GCP

CloudBots can be configured to automatically remediate any rule failures in CloudGuard rulesets for GCP.
When a failure is detected, a GCP function triggers the corresponding CloudBot based on the finding.

The remediation action is run on the entities in the GCP account(s) in which the issue was found.
SendGrid is used to send the remediation output to email.

Intelligence



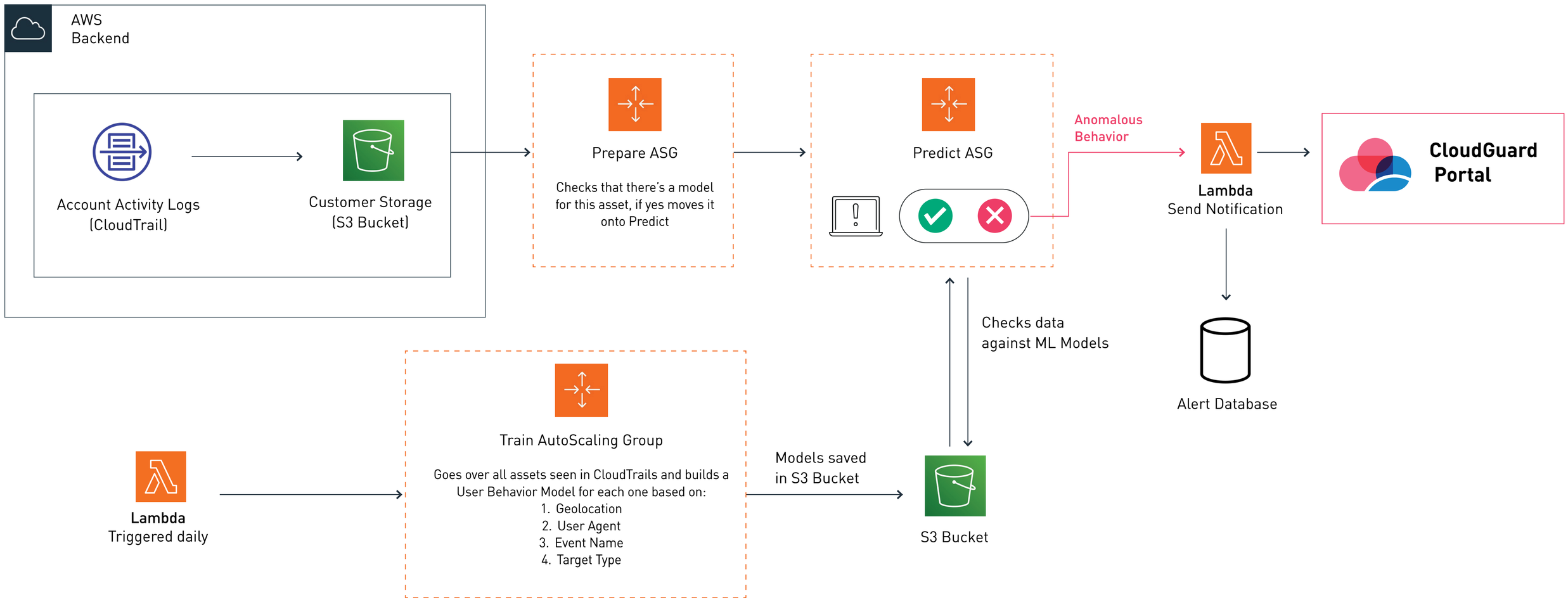


Threat Intelligence and Detection:

Use Case: Log Enrichment and Investigation for AWS

CloudGuard Intelligence draws on two different log sources from AWS accounts to provide in depth log investigative capabilities.

1. Both CloudTrail and Flow Logs are stored in an S3 Bucket in the user's AWS Backend.
2. Those logs are sent from the S3 Bucket to CloudGuard's Simple Queue Service (SQS) and then forwaded along to the CloudGuard Enrichment Pipeline. Here, logs are enriched with information from the CloudGuard inventory and analyzed for anomalous or malicious behaviors.
3. Once logs have undergone an enrichment process, there are several CloudGuard Intelligence Features that can be used. All malicious activity will be alerted through the CloudGuard portal and integrated SIEMS. Users can investigate using interactive visual explorers with the ability to filter with custom queries. All asset history, anomalies, and other relevant information are consolidated into statistic graphs to better understand trends.



Threat Intelligence and Detection:

Use Case: AWS CloudTrail Anomaly Finding

1. In the CloudGuard backend, a lambda is triggered daily to launch the Train AutoScaling Group. Here, a User Behavior Model is built for each asset identified in a user's CloudTrail logs. These ML models are saved in an S3 Bucket in the CloudGuard backend.
2. In the customer's AWS backend, CloudTrail logs are saved to an S3.
3. The Prepare AutoScaling Group checks to see if there is an existing User Behavior Model for each asset identified in the customer's CloudTrail logs. If yes, the asset moves onto the Predict ASG.
4. In the Predict ASG, the asset is compared against it's User Behavior Model to determine if the current activity is considered anomalous or not.
5. If the current activity is anomalous, a lambda is triggered to send a notification to the CloudGuard portal and integrated SIEMS.

Check Point
SOFTWARE TECHNOLOGIES LTD

1

Lambda
Triggered daily

2

Train AutoScaling Group

Goes over all assets seen in FlowLogs and builds a Traffic Behavior Model for each one based on:
1. Geolocation
2. User Agent
3. Event Name
4. Target Type

Models saved in S3 Bucket

3

S3 Bucket

7

Anomalous Behavior

6 Checks data against ML Models

4

Lambda
Triggered hourly

5

Predict Autoscaling Group

Aggregates the FlowLog data from the past hour and saves in separate Database.

Outlier Correlation Autoscaling Group

Further analyzing to prevent false positives and repetitive alerts.

Lambda
Send Notification

CloudGuard Portal

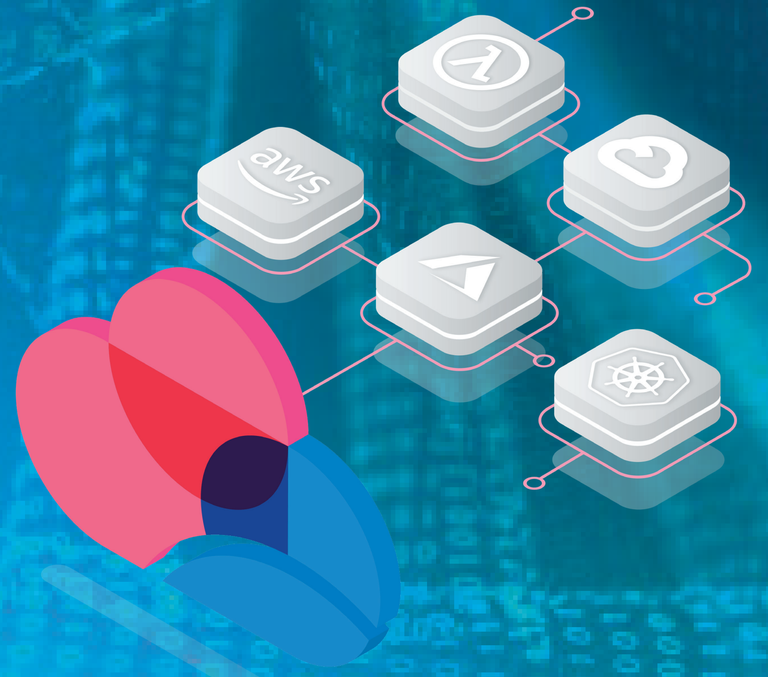
Alert Database

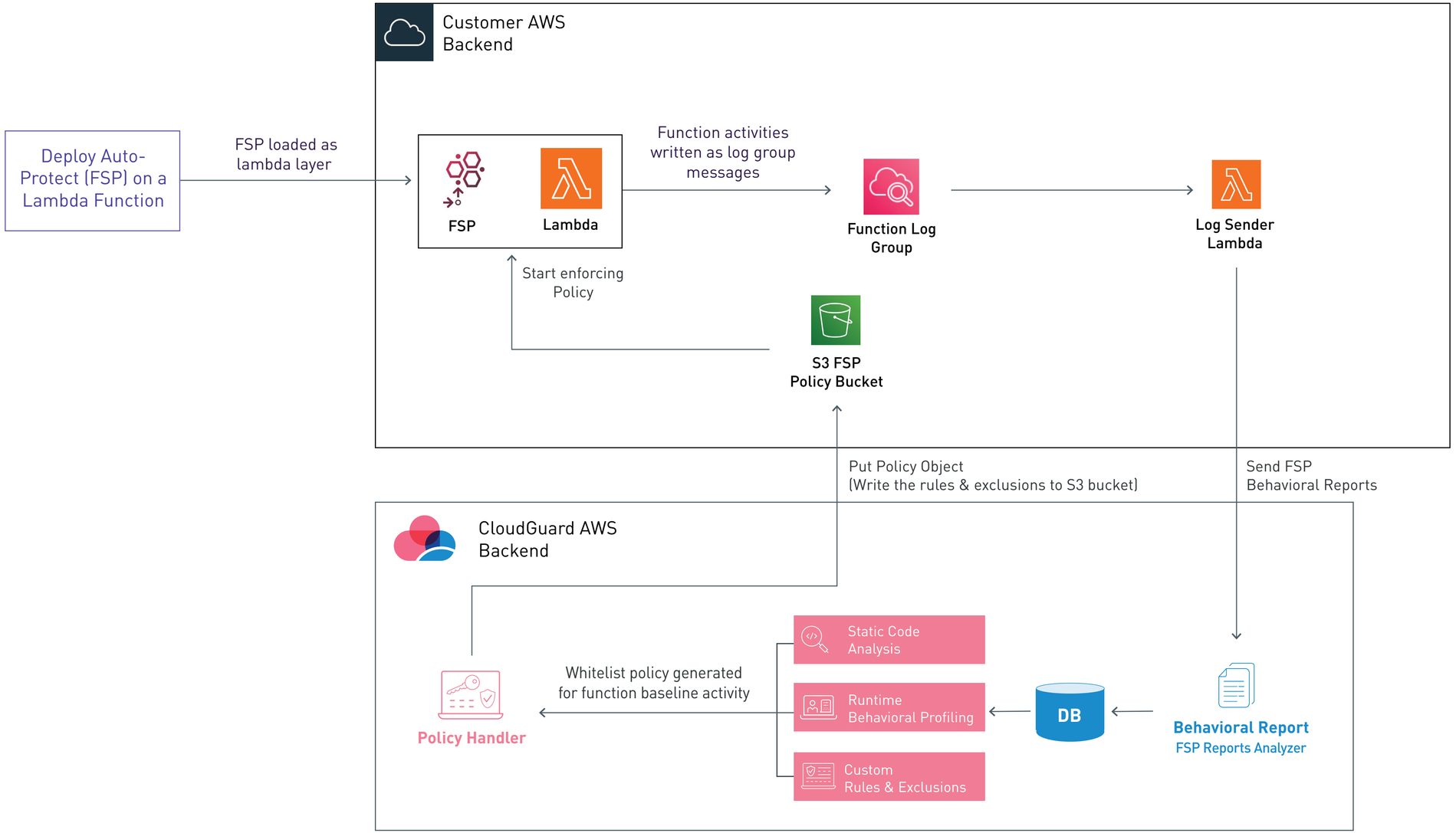
Threat Intelligence and Detection:

Use Case: AWS FlowLog Anomaly Finding

1. In the CloudGuard backend, a lambda is triggered daily to launch the Train AutoScaling Group.
2. In the Train ASG, a Traffic Behavior Model is built for each asset identified in a user's Flow logs.
3. These ML models are saved in an S3 Bucket in the CloudGuard backend.
4. Every hour, a different lambda is triggered to launch the Predict Autoscaling Group.
5. In the Predict ASG, all FlowLog data from the past hour is aggregated and saved
6. The new FlowLog data is checked against the existing ML based Traffic Behavior Models
7. If the new traffic is considered anomalous, the PreCorrelation AutoScaling Group is launched
8. In the PreCorrelation AutoScaling Group, further analysis is done to prevent false positives and repetitive alert. Is there another explanation for this anomaly? For example, maybe this is a weekly update. Have we seen this anomaly in the past before?
9. If the new traffic is determined to be anomalous, a lambda is triggered to send a notification to the CloudGuard portal and integrated SIEMS.

Serverless Security





Serverless Function Runtime Protection

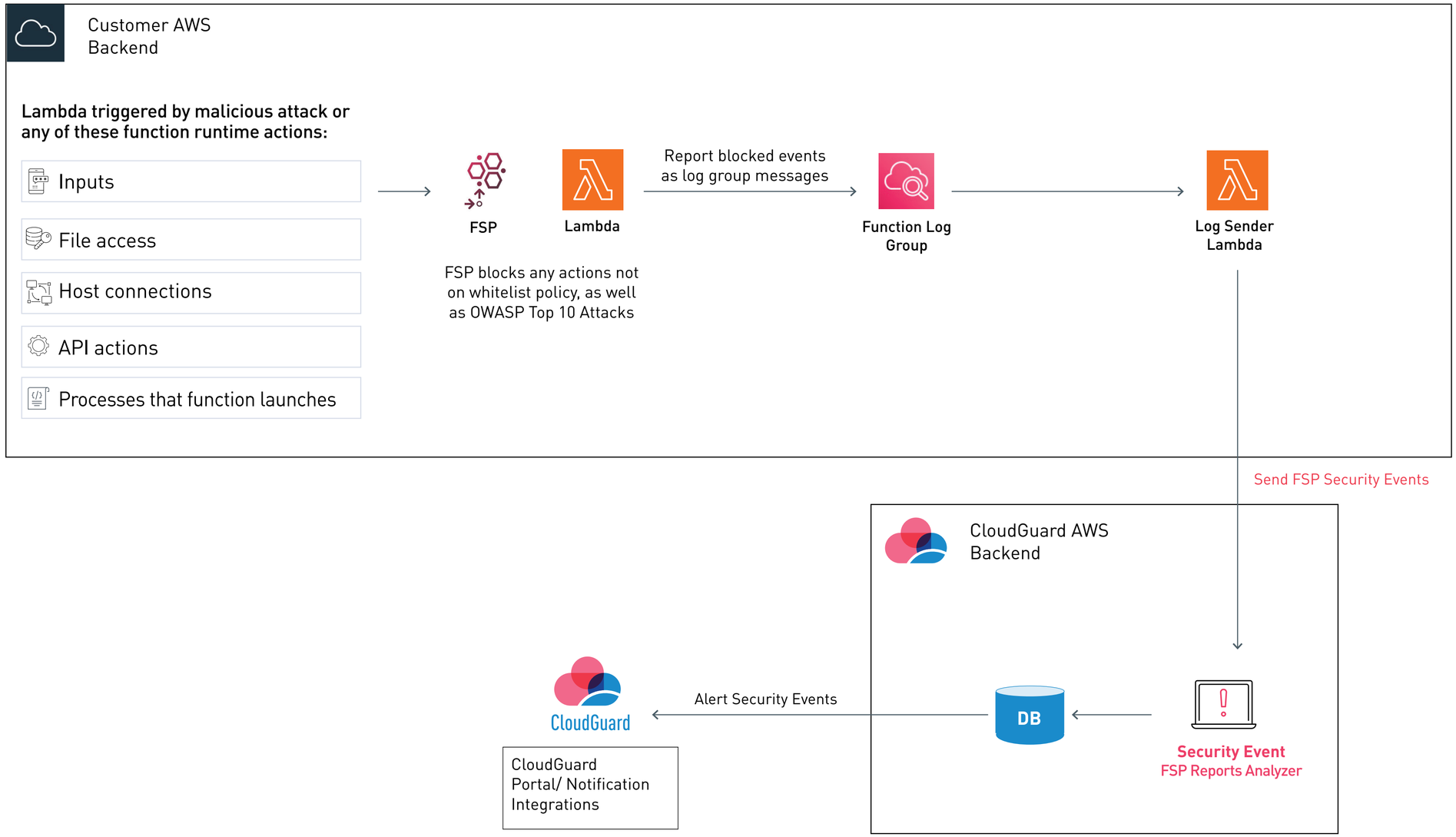
Use Case: Lambda Behavioral Profiling

From within the CloudGuard portal or using ShiftLeft CICD tool, users can deploy Auto-Protect (FSP) on a specific lambda function.
FSP is then loaded as a lightweight lambda layer. It monitors function activities, writes them as log group messages, and prints them to a function log group.

A subscription filter will trigger a lambda function to send these log group messages to the CloudGuard AWS Backend, where they compile into the function's behavioral report.
The backend analyzes this information, as well as static code analysis, in order to generate a whitelist.

This whitelist represents normal, baseline activity for the function and can be viewed and modified with exclusions from within the CloudGuard portal.
This profiling continues until the backend writes a whitelist to the S3 bucket and instructs the Serverless Runtime Protection to begin enforcing it.

The profiling begins when the function is first invoked after being deployed, or after it is changed.
It usually takes a few hundred invocations for the profiling to complete.

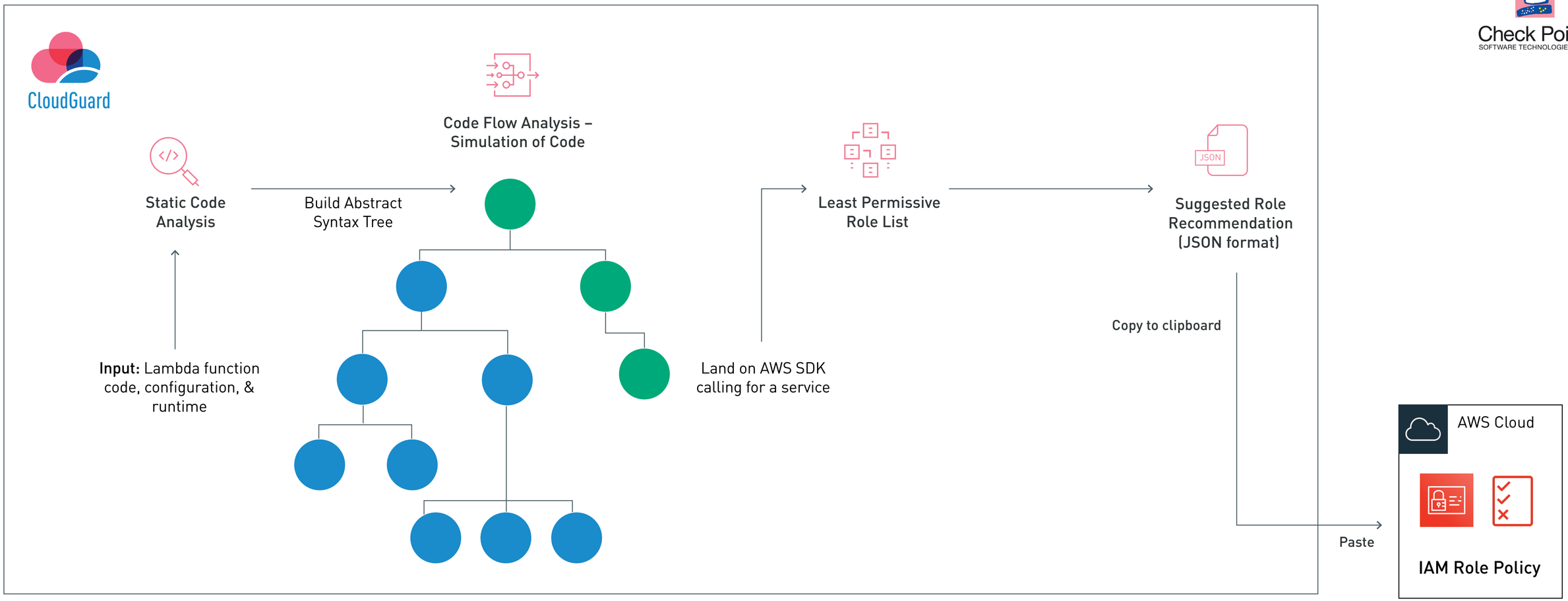


Serverless Runtime Protection

Use Case: Block Threats and Enforce Whitelist Policy

After the behavioral profiling is complete on a lambda function, Auto-Protect can be enabled from the CloudGuard Portal. When instructed by the backend (using the S3 bucket), the Serverless Runtime Protection begins to enforce the whitelist. It monitors activities performed by the function and checks for actions not on the whitelist.

It will report these events as log group messages (which then become CloudGuard Native Alerts).
It can be configured to block these actions as well.
This is determined by user settings when runtime protection is applied to the function.

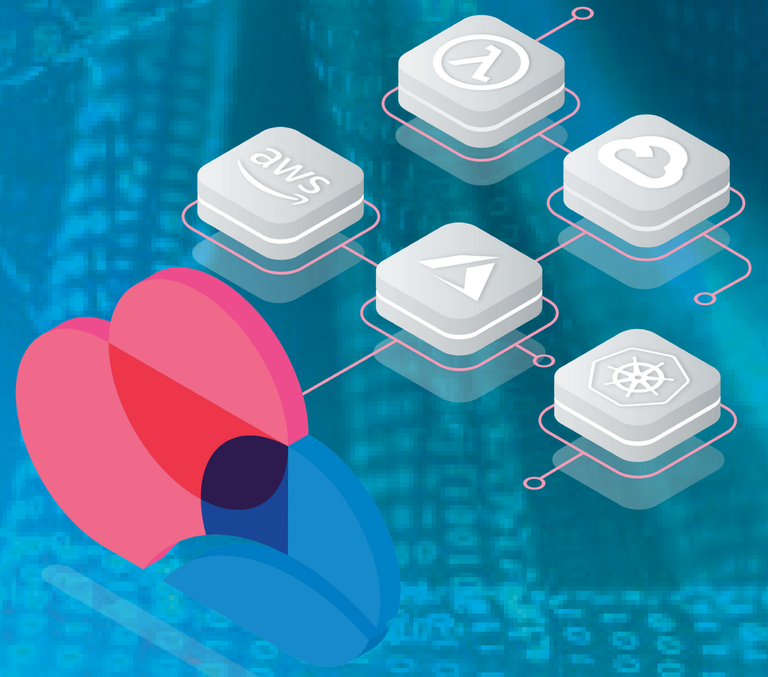


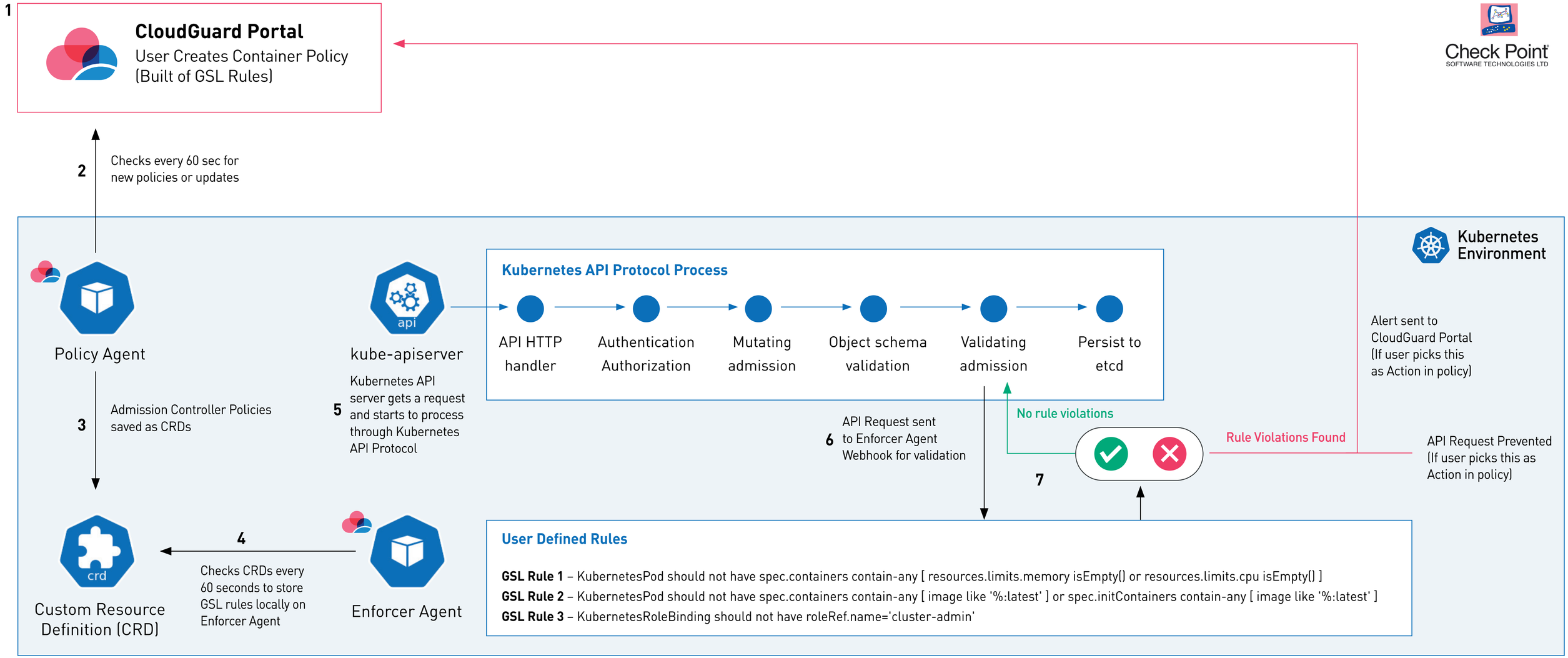
Serverless Static Code Analysis

Use Case – Building a granular whitelist to ensure least permissive

CloudGuard conducts static code analysis on the code and function configurations of a lambda function. It looks for AWS APIs in order to determine what the function actually needs access to. The static code analysis process ends with a Least Privilege Role Recommendation for the lambda function. The code flow analysis does a simulation of the code by building an abstract syntax tree (AST) of all the options and going through them. When it lands on an SDK in AWS that calls for a service, that API is included in the Least Privilege Role. When the analysis is complete, the user is presented with the Suggested Role Remediation in JSON format that can be copied and pasted straight into the AWS IAM Role Policy.

Container Security



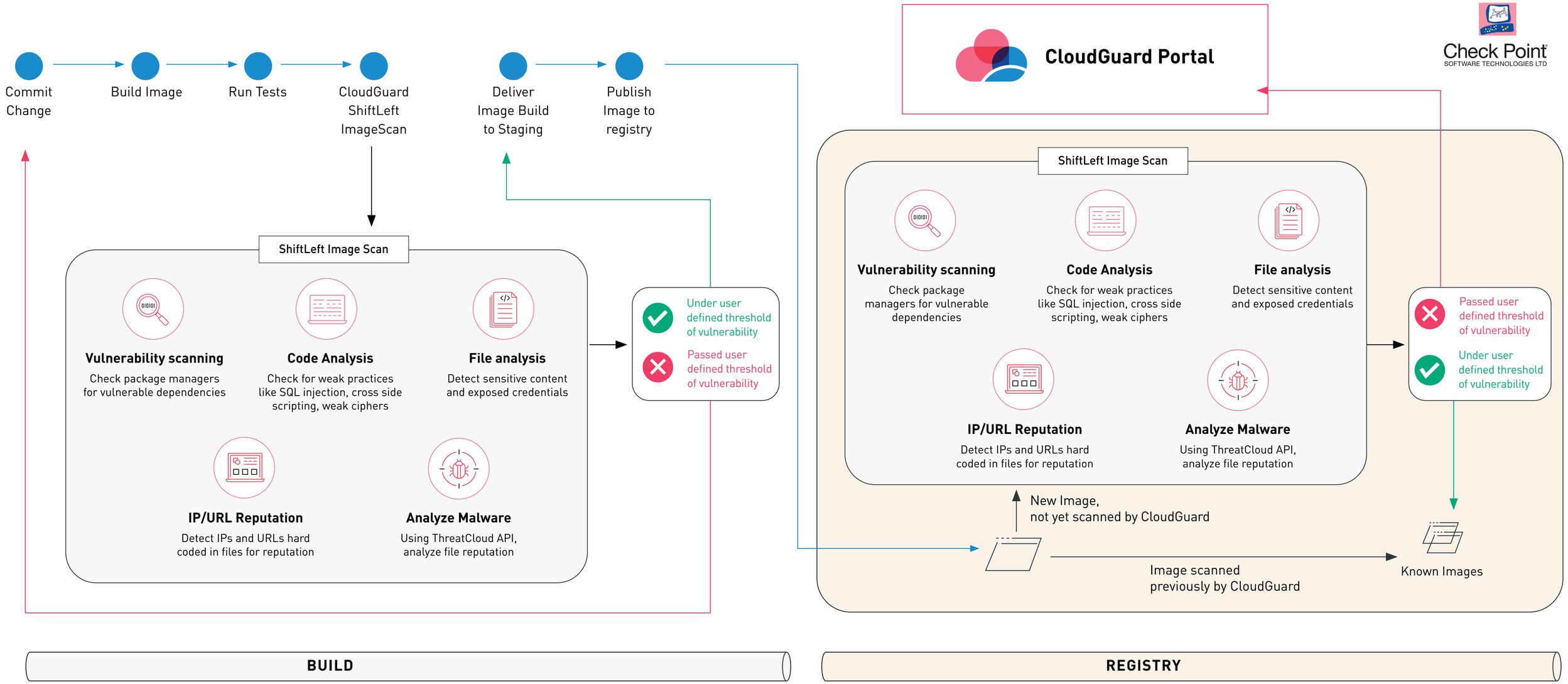


Container Security – User Defined Policies:

Use Case: Enforcing Container Policies using Admission Controller

CloudGuard's Container Security can enforce container policies using the Admission Controller feature of Kubernetes.

1. The user first creates a ruleset/policy using use cases within the CloudGuard portal. CloudGuard will build this policy out of GSL Rules.
2. The CloudGuard Policy agent deployed within a user's Kubernetes environment checks every 60 seconds for new policies or updates.
3. The Policy agent saves the policies as Custom Resource Definitions (CRDs).
4. The Enforcer Agent checks CRDs every 60 seconds so it can have the GSL Rules locally.
5. When the Kubernetes API server gets a request it begins the API Protocol Process, going through the stages in the diagram. In the validating admission step, the API server sends the request to be validated by the Enforcer agent.
6. The Enforcer agent compares the request against the user defined rules. If it violates the rules, the API request is either prevented or an alert is sent to the CloudGuard portal. The user can decide which action they'd like the Enforcer Agent to take within the CloudGuard portal ahead of time.

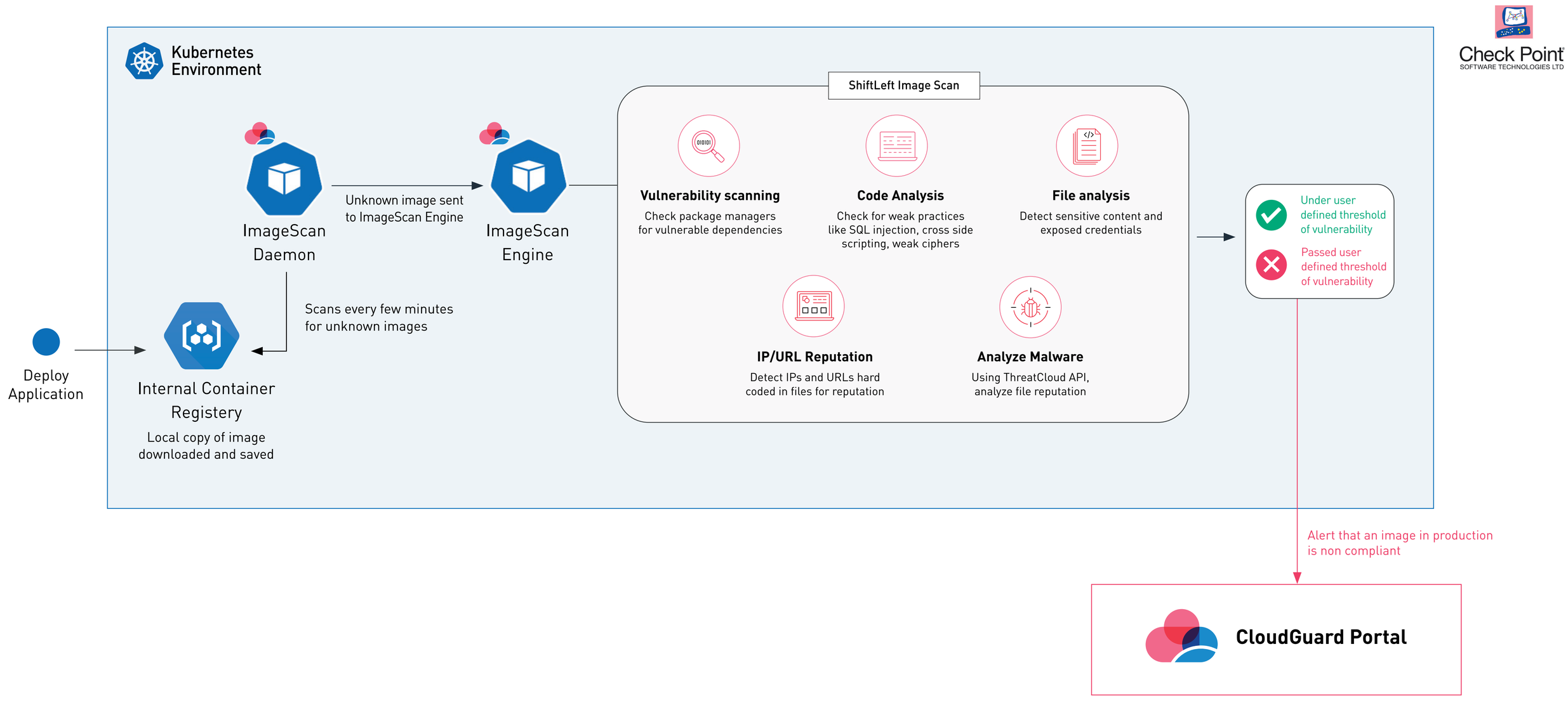


Container Security – Image Assurance:

Use Case: Continuous Vulnerability Assurance Throughout Image Lifecycle – Pre Production

CloudGuard's container protection will check your container images for vulnerabilities in all stages during pre-production.
CloudGuard's ShiftLeft ImageScan can be inserted into the BUILD Stage of a CICD Pipeline. Several different scans are employed to ensure the image is vulnerability free and will not expose any sensitive content. Users will define their threshold of vulnerability. If an image surpasses this, it will be blocked. Users will need to fix the issues and then recommit before proceeding. Otherwise, the image build pipeline can continue.

Within the Registry, any new image that has yet to be scanned will undergo the same ImageScan process. If the image surpasses the user defined threshold for vulnerability, an alert is sent to the CloudGuard portal. If the image is clean, it will be put into Known Images, from which it can be deployed into production.
Users can utilize Admission Controller enabled policies to allow only compliant images to be deployed into production. Compliance is defined by the user within the CloudGuard portal.

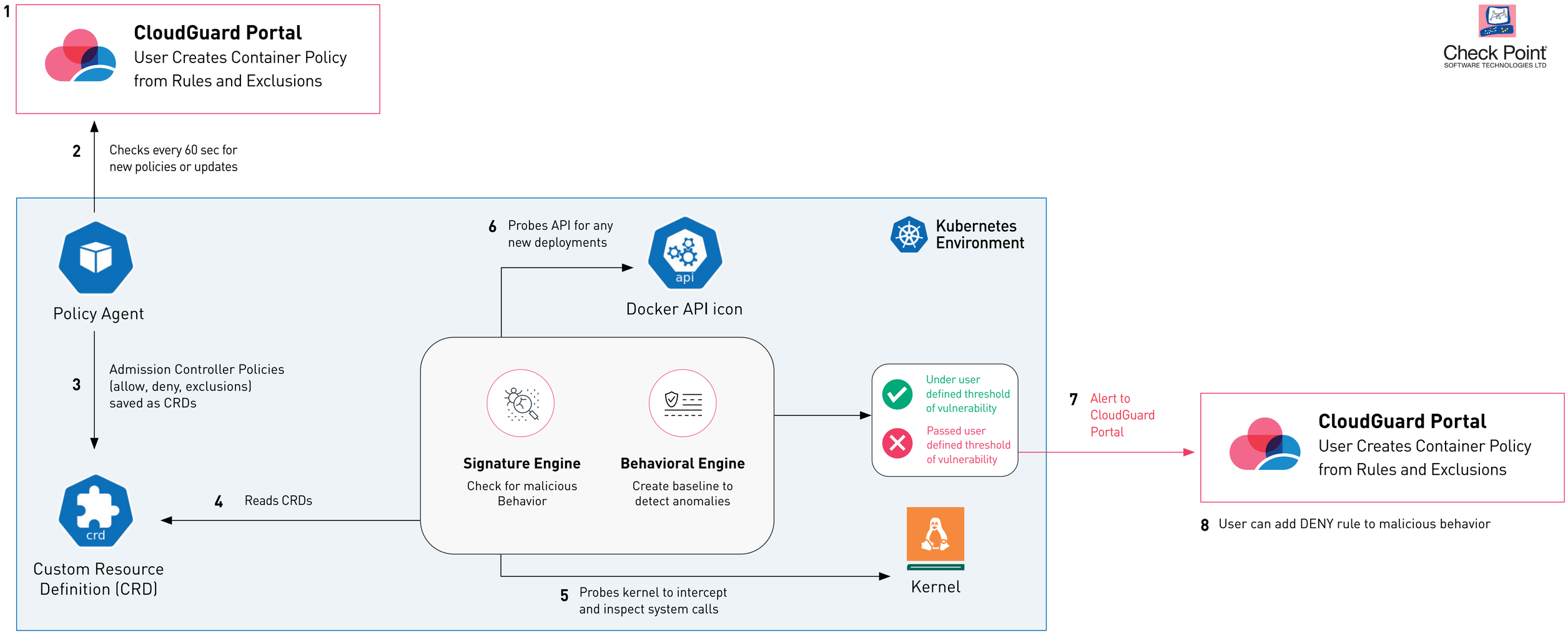


Container Security – Image Assurance:

Use Case: Continuous Vulnerability Assurance Throughout Image Lifecycle – Production

Once an application is deployed, a local copy of the image is downloaded and saved into the internal container registry, which exists in every node. CloudGuard's ImageScan Daemon scans the local container registry every few minutes for any unknown images.

If the ImageScan Daemon identifies an unknown image, it passes that image to the ImageScan engine agent. Inside the ImageScan Engine agent (1 per cluster by default), several different scans are employed to ensure the image is vulnerability free and will not expose any sensitive content. Users will define their threshold of vulnerability. If an image surpasses this, an alert is sent to the CloudGuard portal that an image in production is non-compliant.

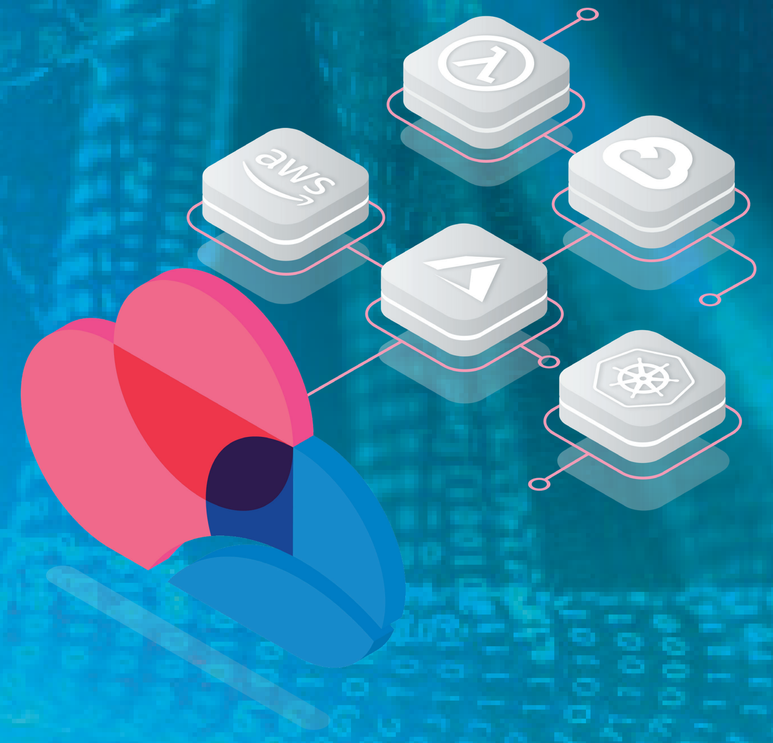


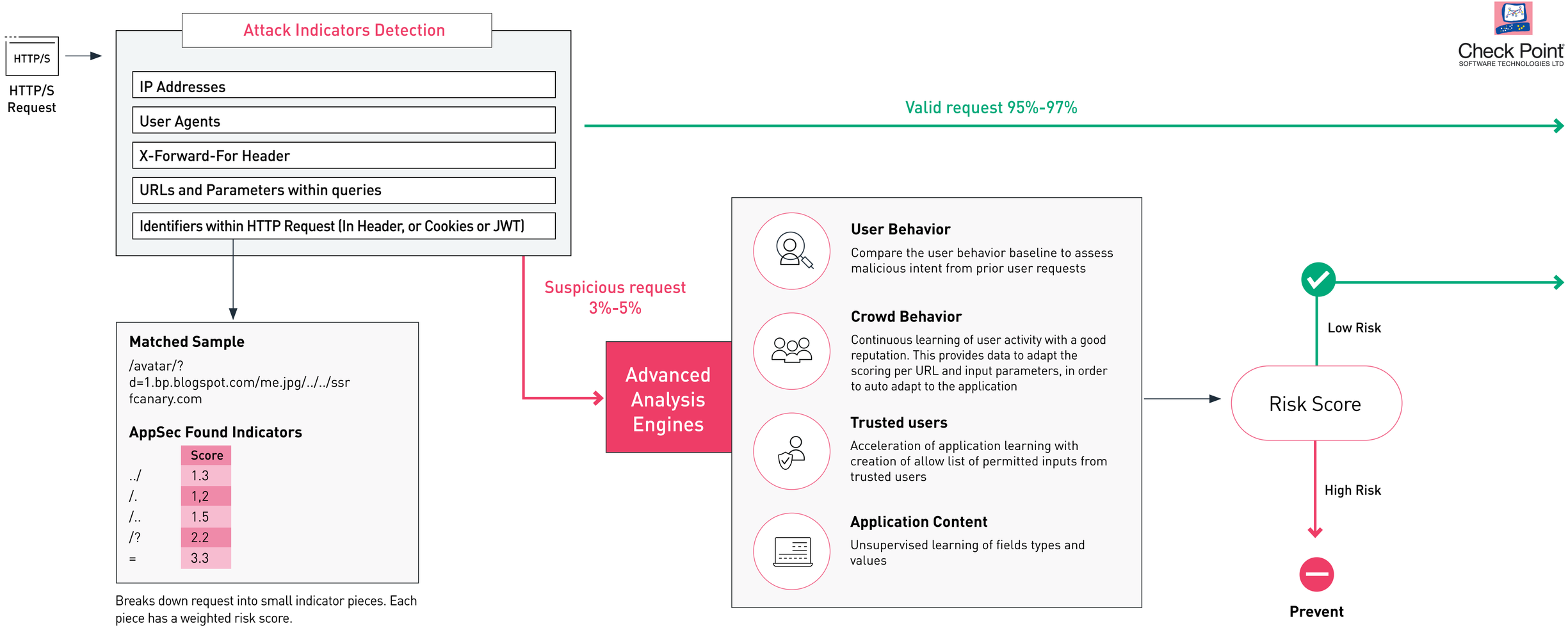
Container Security

Use Case: Runtime Protection

1. In the CloudGuard portal, users can create a policy for containers. They choose from different use cases they want to enforce and a policy is automatically built of GSL rules.
2. The Runtime Policy Agent, a single-replica Deployment, is responsible for retrieval of configuration and policies that the user configures from the CloudGuard backend. It checks every 60 seconds for new policies or updates
3. Policies have the options of allow, deny, and exclusions. They are saved by the CloudGuard Policy Agent as local Custom Resource Definitions (CRD). Only the Policy Agent has permissions to modify the CRDs, and only the Policy and Runtime Daemon Agents have permissions to read the CRDs. Access to the CRDs, including read permissions such as get and list, should be restricted, because they contain information on the cluster security configuration.
4. The Runtime Daemon is a Daemon set which reads the CRDs in order to check system calls and new deployments against the user defined policies.
5. The Runtime Daemon probes the kernel to intercept and inspect the system calls.
6. The Runtime Daemon probes the API for any new deployments in order to enrich kernel events with container and Pod Group details
7. The Runtime Daemon has two Runtime Protection Engines. The Signature engine checks for any malicious behavior. It compares the observed behavior of a workload with known signatures that potentially indicate malicious behavior, for example, execution of processes associated with crypto-mining software. The Behavioral Engine detects anomalies in behavior compared to a baseline profile created during a dedicated profiling phase, for example, execution of sub-process that do not occur during regular workload operation, which may indicate an RCE attack. If any malicious behavior or anomalies are detected, an alert is sent to the CloudGuard Portal.
8. From within the CloudGuard portal, users can add a DENY rule to any alert for malicious behavior, with the option of either making the pod restart or killing the container every time this behavior is seen again.

Application Security



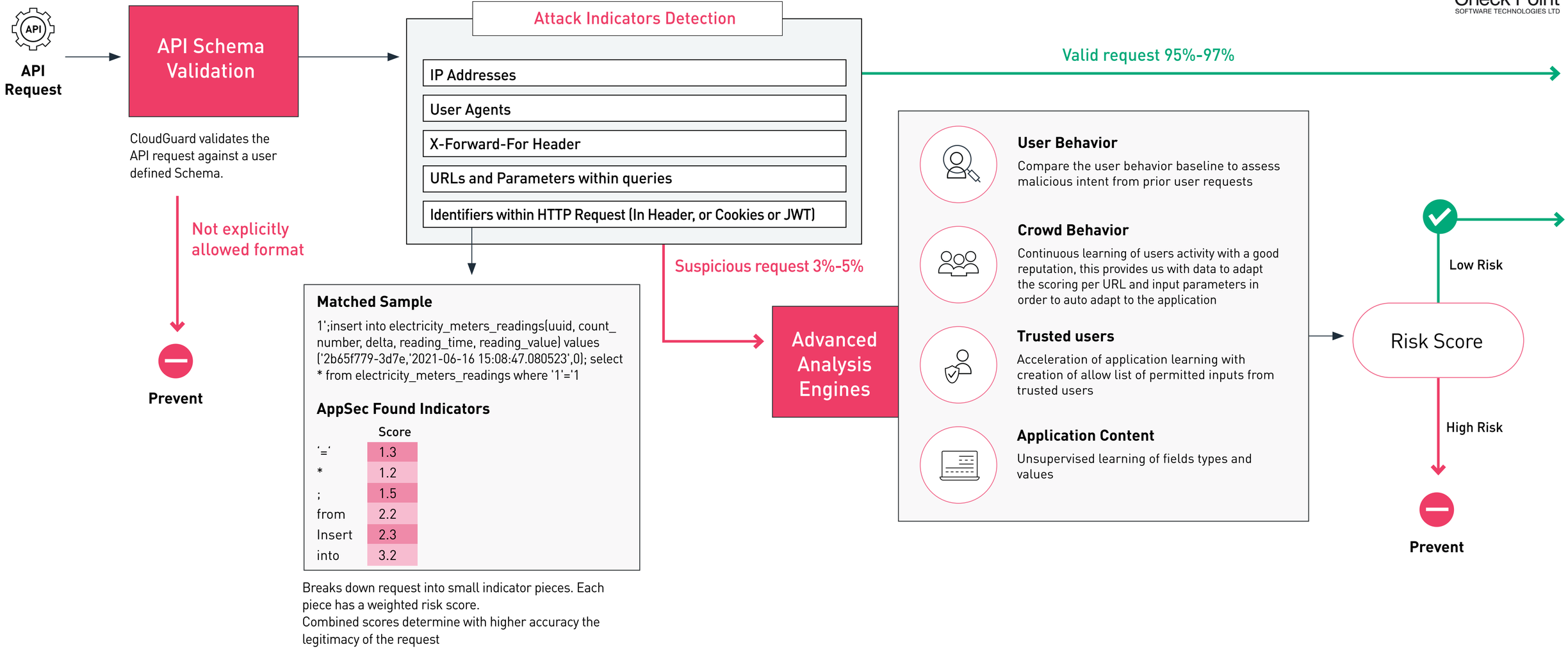


Breaks down request into small indicator pieces. Each piece has a weighted risk score.
Combined scores determine with higher accuracy the legitimacy of the request

Web Application Security - HTTP(S) Requests:

Use Case: Accurate Web Application Protection Utilizing Machine Learning Engines

CloudGuard AppSec analyzes HTTP/S requests for validity. When a connection arrives, AppSec will try to isolate and understand the identity source by differentiating between human users, machine users, and APIs.
The requests are broken down into identifiers such as IP addresses, user agents, X-forward-for-header, and URLs and parameters within queries. These are analyzed against ThreatCloud, the world's largest cloud based cybersecurity intelligence sharing system.
AppSec will also break down an HTTP request into smaller indicator pieces. Using an on-going offline supervised training process, a base model will provide each indicator with a score that indicates likelihood of being part of an attack. This score is combined to determine with higher accuracy the legitimacy of the request.
If a request is determined to be potentially suspicious, it will move onto Advanced Analysis Engines. Utilizing machine learning, these engines together will provide a an accurate risk score. If a request is determined to be risky, it can confidently be prevented. Otherwise, it will be allowed.

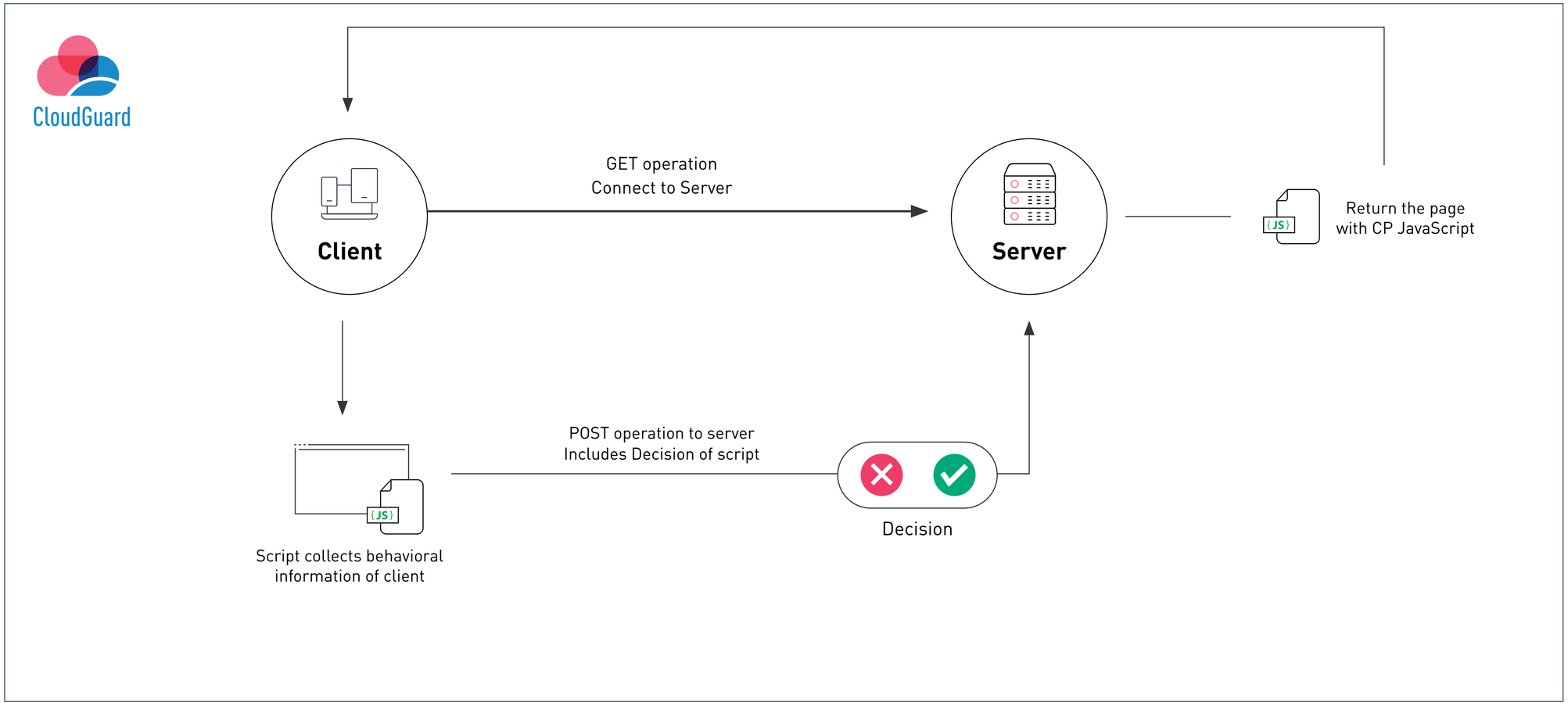


Web Application Security – API Requests:

Use Case: Accurate API Protection Utilizing Machine Learning

CloudGuard AppSec analyzes API requests for validity.

1. The API request is first validated against a user defined schema to make sure it is an explicitly allowed format. If not, the request is prevented.
2. Next, the request is broken down into identifiers such as IP addresses, user agents, X-forward-for-header, and URLs and parameters within queries. These are analyzed against ThreatCloud, the world's largest cloud based cybersecurity intelligence sharing system.
3. AppSec will also break down an API request into smaller indicator pieces. Using an on-going offline supervised training process, a base model will provide each indicator with a score that indicates likelihood of being part of an attack. This score is combined to determine with higher accuracy the legitimacy of the request.
4. If a request is determined to be potentially suspicious, it will move onto Advanced Analysis Engines. Utilizing machine learning, these engines together will provide a an accurate risk score. If a request is determined to be risky, it can confidently be prevented. Otherwise, it will be allowed.

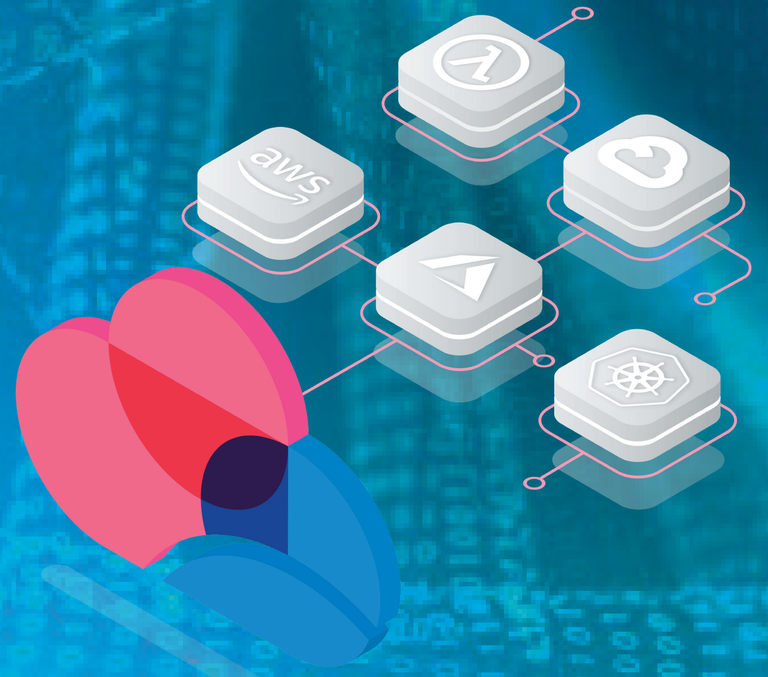


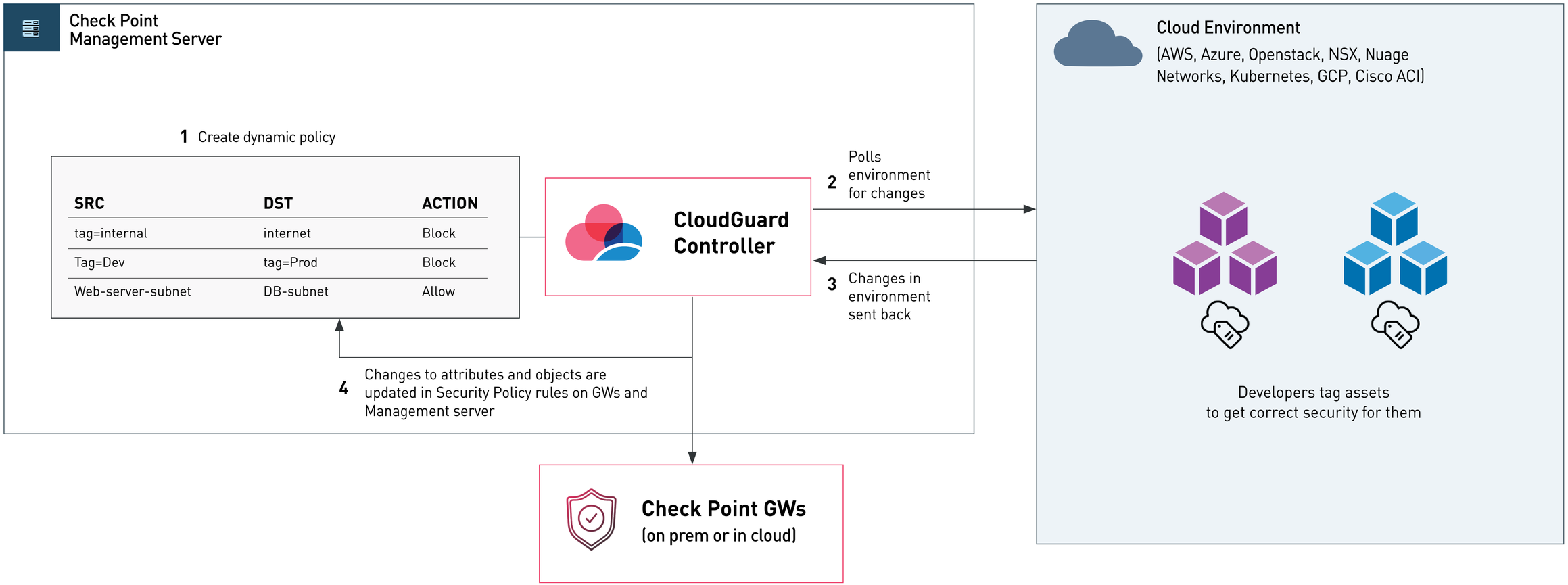
Application Security

Use Case – Prevent Bot Attacks while allowing Legitimate Traffic

CloudGuard AppSec utilizes Client Side Behavioral Analysis to distinguish human behavior from malicious bots. Once a client connects to a server, they perform a GET operation. They receive from the server the page which includes a Javascript developed by Check Point that is injected into the browser of the client. This script collects behavioral information from the client. When the client performs a POST operation to the server, it will include the decision of this script, which defines if the request originated from a bot or a human.

Network Security





Network Security - Automatically Adaptive Security Policies:

Use Case: Security Policy automatically updated to cloud environment changes

CloudGuard Network Security allows for truly dynamic security policies with automatic updates of cloud asset's metadata.

1. CloudGuard Controller is utilized to create a connection between the management server and the cloud environment. This allows the user to build a dynamic policy utilizing cloud objects or tags imported directly from the cloud environment.
2. CloudGuard Controller continually polls the cloud environments for any changes.
3. Any changes to attributes, such as a new IP address, new instance created or deleted, or tags updated, will be updated within the management server within a minute of the change.
4. Security policy rules within the management server are automatically updated with these changes, which propagate out to the GWs for enforcement.

This dynamic enforcement of network security allows for a separation of responsibilities between the Security and DevOps teams. Once it is decided how each type of asset should be tagged, Security teams can build a corresponding ruleset. All DevOps needs to do is tag the assets correctly for security to automatically follow.

Check Point
SOFTWARE TECHNOLOGIES LTD

Create Data Center Query Object

Data Center Query

azure_dcq_production_staging

Enter Object Comment

Data Centers

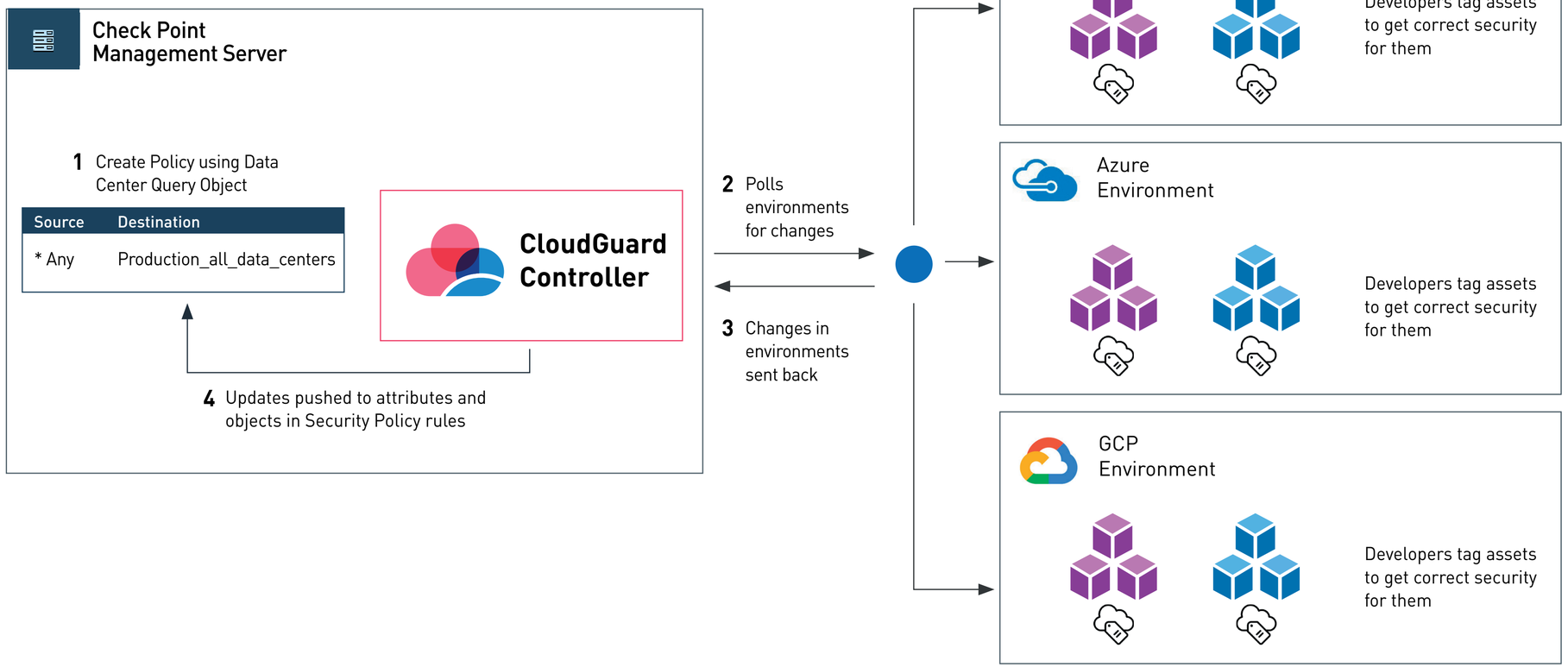
- ○ All Data Centers
- ◉ Specific Data Centers

Name	Comments
azure_production	
azure_staging	

Query Rules

Key	Value(s)
azure_production	production; staging
azure_staging	Virtual Machine, Subnet

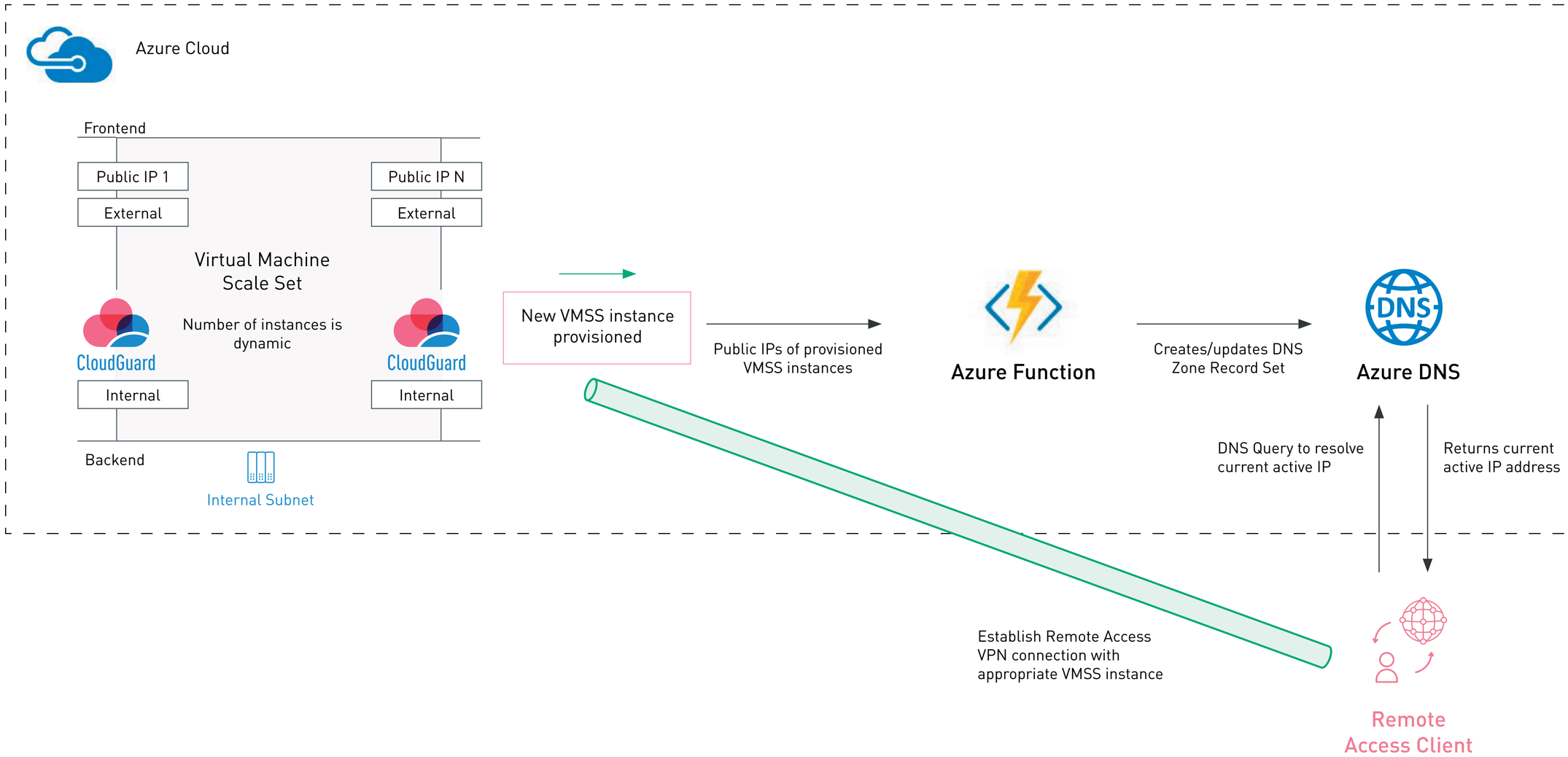
OK Cancel



Network Security - Cloud Agnostic Policies:

Use Case: Data Center Agnostic Rules in Policy

CloudGuard Network Security lets users create a cloud environment agnostic object to automatically enforce a policy for new accounts across cloud providers.
0. First, users must create a Data Center Query object within the SmartConsole management portal. There, they define which data centers (cloud accounts) should be included in the object. Query rules allow users to be more specific - you can specify certain tags or types within a data center that this Data Center Query object should apply to. Query rules have a logical AND between them and logical OR to values on the same rule.
1. The Data Center Query object is utilized within a policy
2. CloudGuard Controller establishes a connection between the management server and the data centers (cloud environments). It continually polls these environments for any changes.
3. Any changes to attributes, such as a new IP address, new instance created or deleted, or tags updated, will be updated on the security gw in short time
4. When new cloud accounts are added, a simple policy push is all that is required for the security policy utilizing Data Center Query Object to automatically apply security. No need to change the policy.



Network Security

Use Case: Scalable Remote Access VPN (Azure)

1 Azure function gets Public IPs of VMSS instances whose provision process is finished.
2 Azure function creates, in case it does not exist, or updates a DNS Zone Record Set with the VMSS Instances' public IPs.
3 Remote Access VPN client runs a DNS query to resolve the current active IP addresses.
4 Azure DNS returns the current active IP addresses.
5 Client does a load-share mechanism on the resolved IP list and establishes Remote Access VPN connection with the appropriate VMSS instance. It then gets access to internal resources.