



BMW Group accelerates cloud optimization using Amazon Bedrock

Dr. Jens Kohl

Head of Offboard Architecture
BMW Group

Christian Müller

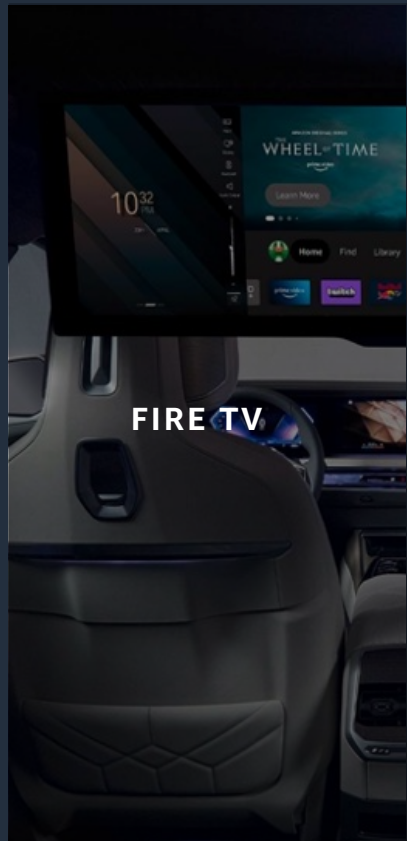
Principal Solutions Architect
AWS

Agenda

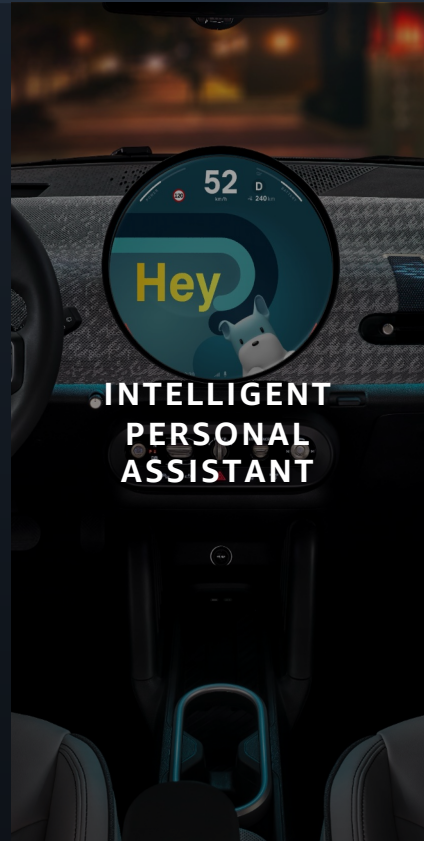
- History of Connected Vehicle Backend
- Cloud Governance Tool
- In-Console Cloud Assistant



BMW offers premium digital customer experiences



FIRE TV



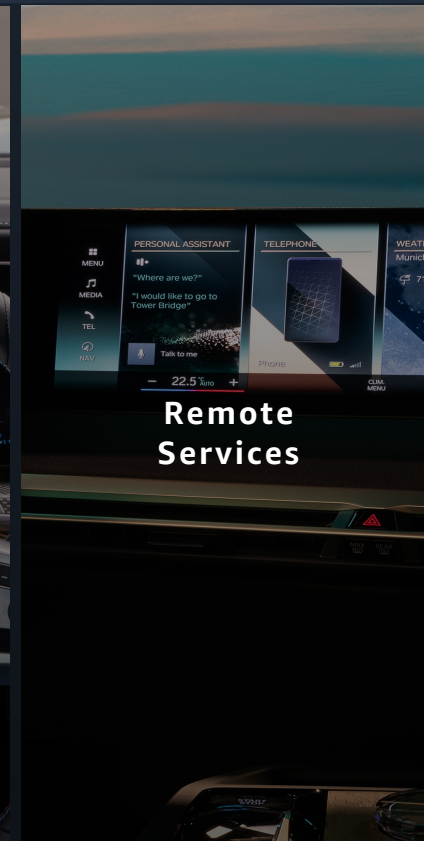
**INTELLIGENT
PERSONAL
ASSISTANT**



**ROUTE OPTIMIZED
FOR CHARGING**



**Real Time
Traffic Information**



**Remote
Services**



**CONNECTED VEHICLE
& AI PLATFORM**

BMW ConnectedDrive in numbers

22M+

CONNECTED
VEHICLES

7M+

FULLY
UPGRADEABLE
VEHICLES

12B+

REQUESTS
PER DAY

1300+

MICROSERVICES

145TB

DATA TRAFFIC
PER DAY

99.95%

RELIABILITY

Expected to triple over the next two years

22M+

CONNECTED
VEHICLES

7M+

FULLY
UPGRADEABLE
VEHICLES

12B+

REQUESTS
PER DAY

1300+

MICROSERVICES

145TB

DATA TRAFFIC
PER DAY

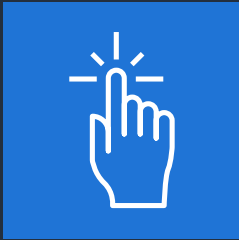
99.95%

RELIABILITY

Our measures to start automating the flywheel



Gain insights

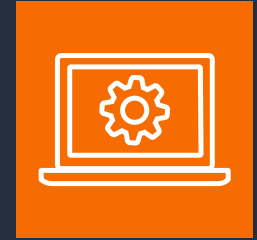


Identify actionable items

Learn



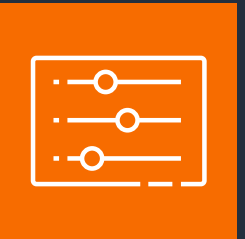
Build



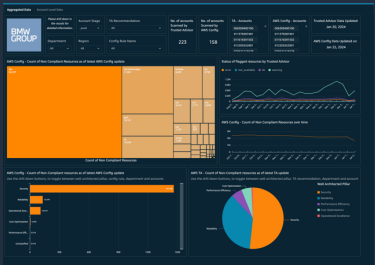
Optimize accounts



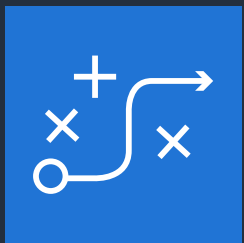
Adding documentation and remedies to rules



Prioritize rules



Visualize prioritized findings

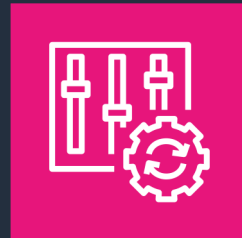


Identify additional rules

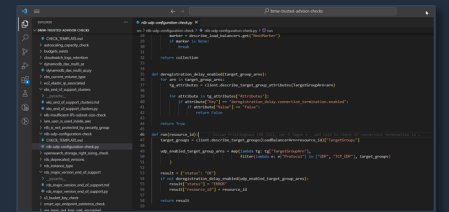


AWS Trusted Advisor

Measure

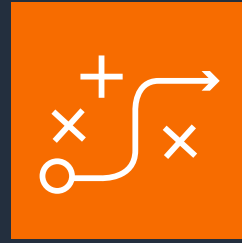
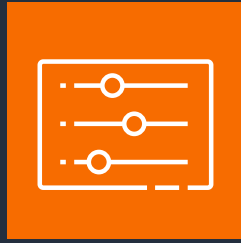


AWS Config



Extend ruleset

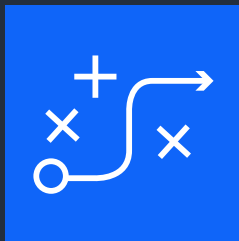
Step 1: Enrich rule evaluation data



Nr.	Config Rule Name	Well Architected Pillar	Priority	Config Rule Description	Link / Source	Rule Source	State
03	config-rule-eks-optimized-instance	Performance Efficiency	Medium	Checks if EBS optimization is enabled for your EC2 instances that can be EBS-optimized. The rule is NON_COMPLIANT if EBS optimization is not enabled for an EC2 instance that can be EBS-optimized.	https://docs.aws.amazon.com/config/latest/developerguide/eks-optimized-instance.html	MANAGED	IN_PROD
04	config-rule-eks-resources-protected-by-backup-plan	Operational Excellence	High	Checks if Amazon Elastic Block Store (Amazon EBS) volumes are protected by a backup plan. The rule is NON_COMPLIANT if the Amazon EBS volume is not covered by a backup plan.	https://docs.aws.amazon.com/config/latest/developerguide/eks-resources-protected-by-backup-plan.html	MANAGED	IN_PROD
05	config-rule-rds-aurora-availability-level	Reliability	High	Rule implements availability levels defined in https://atc.bmwgroup.net/confluence/display/CESCC/Connected+Paas+-+Availability+of+application+services+and+application+resources	config-rule-rds-aurora-availability-level	CUSTOM	IN_PROD
06	config-rule-rds-postgresql-availability-level	Reliability	High	Rule implements availability levels defined in https://atc.bmwgroup.net/confluence/display/CESCC/Connected+Paas+-+Availability+of+application+services+and+application+resources	config-rule-rds-postgresql-availability-level	CUSTOM	IN_PROD
18	lambda_dlg_check lambda-dlg-check	Reliability	Medium	Checks whether the AWS Lambda function is configured for a Dead Letter Queue(DLQ). The rule is NON_COMPLIANT if the Lambda function is not configured for DLQ.	https://github.com/awslabs/aws-config-rules/tree/master/system/LAMBDA_DLQ_CHECK	CUSTOM	IN_PROD

Store enriched rules in knowledge base

Step 3: Identify new checks/rules and extend ruleset



Identify additional rules from best practices, post mortems,



```
def deregistration_delay_enabled(target_group_arns):  
    for arn in target_group_arns:  
        tg_attributes = client.describe_target_group_attributes(TargetGroupArn=arn)  
        for attribute in tg_attributes["Attributes"]:  
            if attribute["Key"] == "deregistration_delay.connection_termination.enabled":  
                if attribute["Value"] == "false":  
                    return False  
    return True  
  
def run(resource_id):  
    target_groups = client.describe_target_groups(LoadBalancerArn=resource_id)["TargetGroups"]  
    udp_enabled_target_group_arns = map(lambda tg: tg["TargetGroupArn"],  
                                       filter(lambda e: e["Protocol"] in ["UDP", "TCP_UDP"], target_groups))  
    result = [{"status": "OK"}]  
    if not deregistration_delay_enabled(udp_enabled_target_group_arns):  
        result["status"] = "ERROR"  
        result["resource_id"] = resource_id  
    return result
```

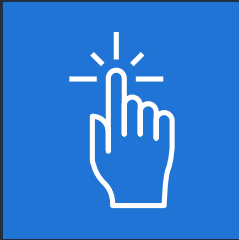
Extend ruleset

<https://aws.amazon.com/blogs/mt/aws-config-rule-development-kit-library-build-and-operate-rules-at-scale/>

Our measures to start automating the flywheel



Gain insights



Identify actionable items

Learn



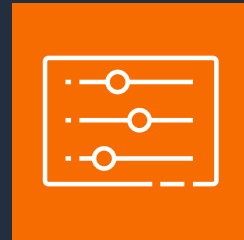
Build



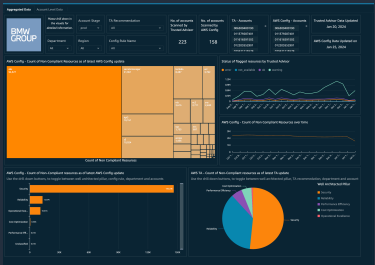
Optimize accounts



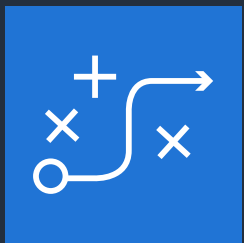
Adding documentation and remedies to rules



Prioritize rules



Visualize prioritized findings

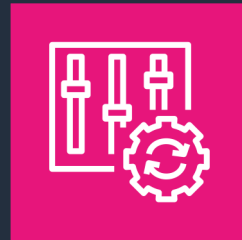


Identify additional rules

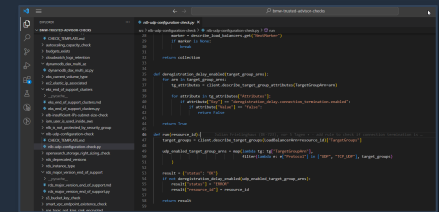


AWS Trusted Advisor

Measure



AWS Config



Extend ruleset

Accelerating the Flywheel even more

Learn

Build



Measure

please explain remedies for Low Utilization Amazon EC2 Instances

Here are the recommendations:

- Stop or terminate underutilized EC2 instances to optimize costs.
- Use Auto Scaling to scale EC2 fleet based on demand.
- Change the instance size to optimize performance and costs.

Explain findings and remedies

```
import boto3

client = boto3.client('ec2')

# Insert your Instance ID here
my_instance = instance_id

# Check whether the instance is running
response = client.describe_instance_status(InstanceIds=[my_instance])

if len(response['InstanceStatuses']) == 0:
    print('There are no active instances')
elif response['InstanceStatuses'][0]['InstanceState']['Name'] == 'running':
    print('The instance need to be stopped prior changing the instance type')

# Stop the instance
client.stop_instances(InstanceIds=[my_instance])
waiter=client.get_waiter('instance_stopped')
waiter.wait(InstanceIds=[my_instance])

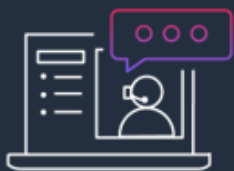
# Change the instance type
client.modify_instance_attribute(InstanceId=my_instance, Attribute='instanceType', Value=instance_type)

# Start the instance
client.start_instances(InstanceIds=[my_instance])
```

Generate code

Overview of the bot

Use-Cases



Generic Question Answering



Account Health Reporting



Issue Resolution



Deploying Changes

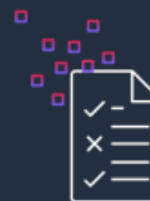
Data Sources



Well-Architected Framework and Service Documentation

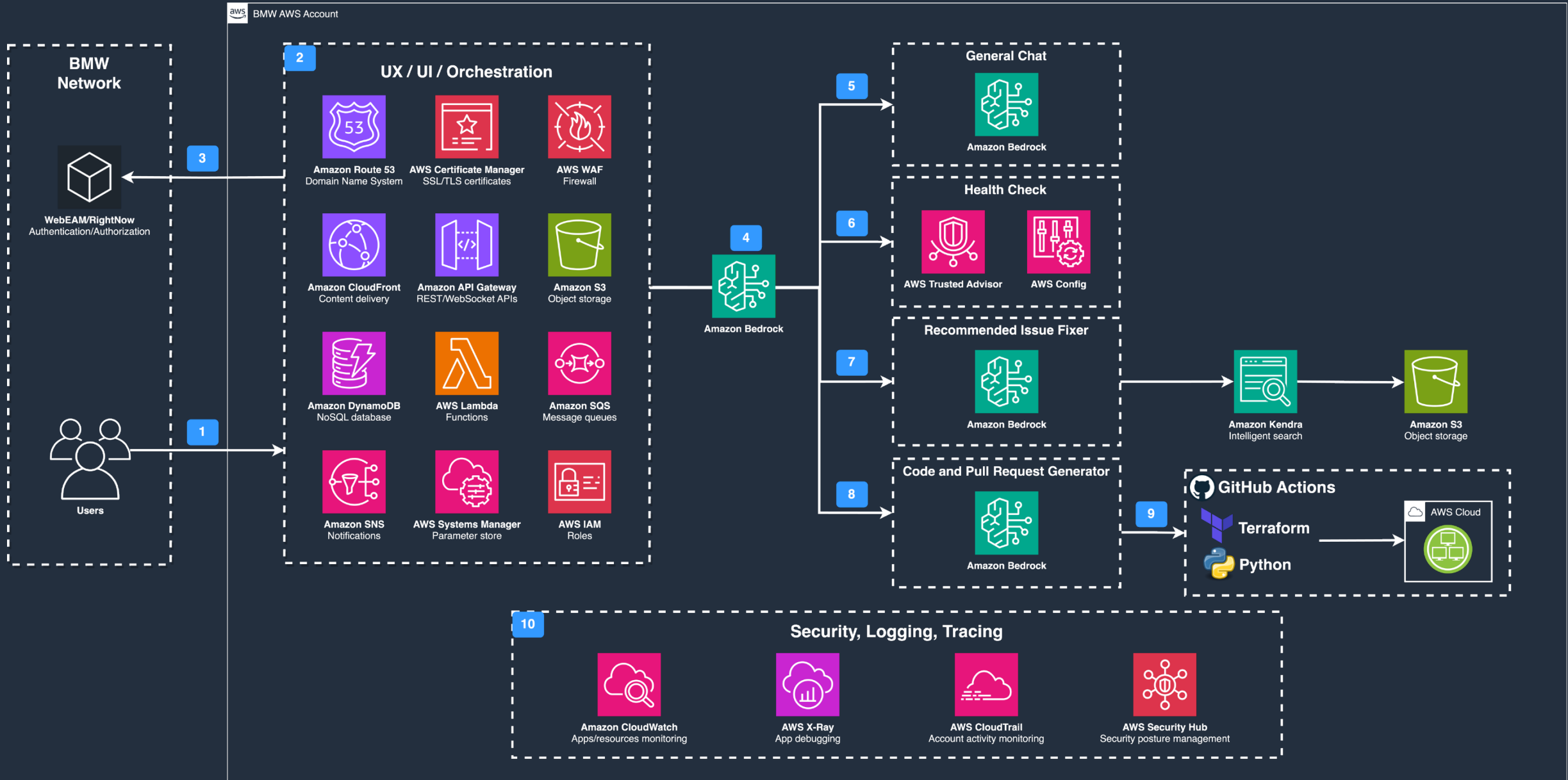


Report from Trusted Advisor

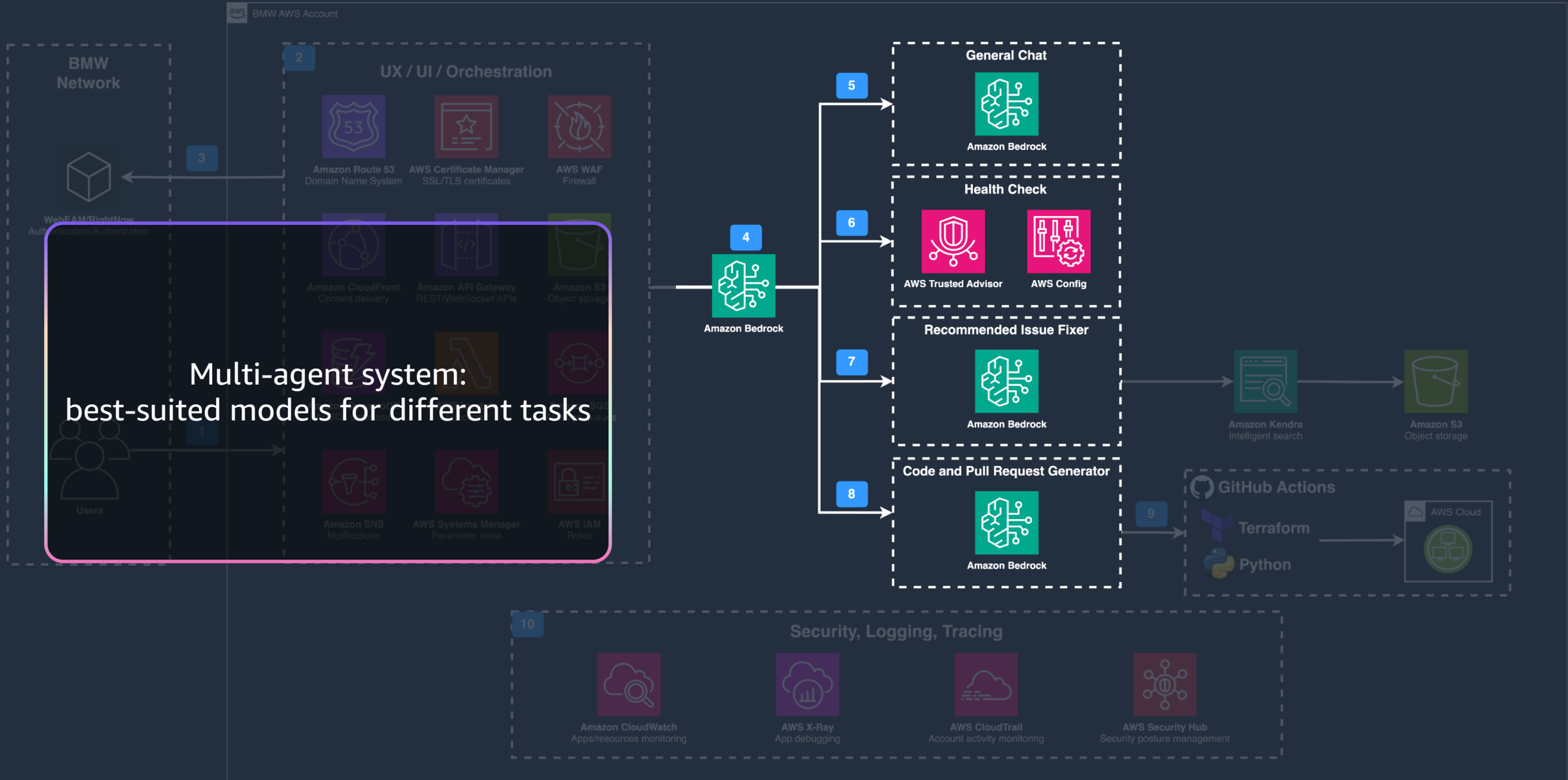


AWS Config Checks

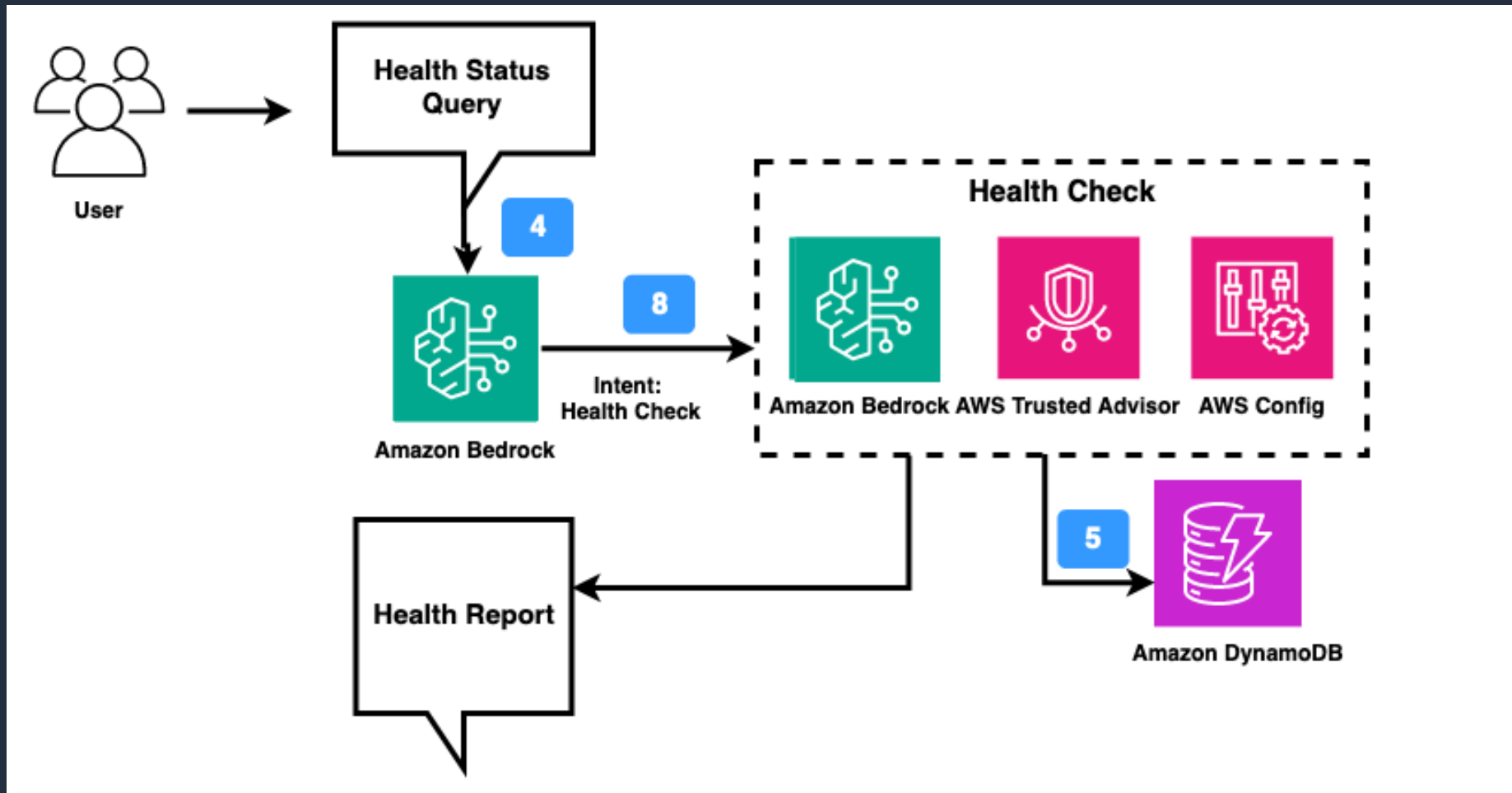
BMW GROUP ACCELERATES CLOUD OPTIMIZATION USING AMAZON BEDROCK



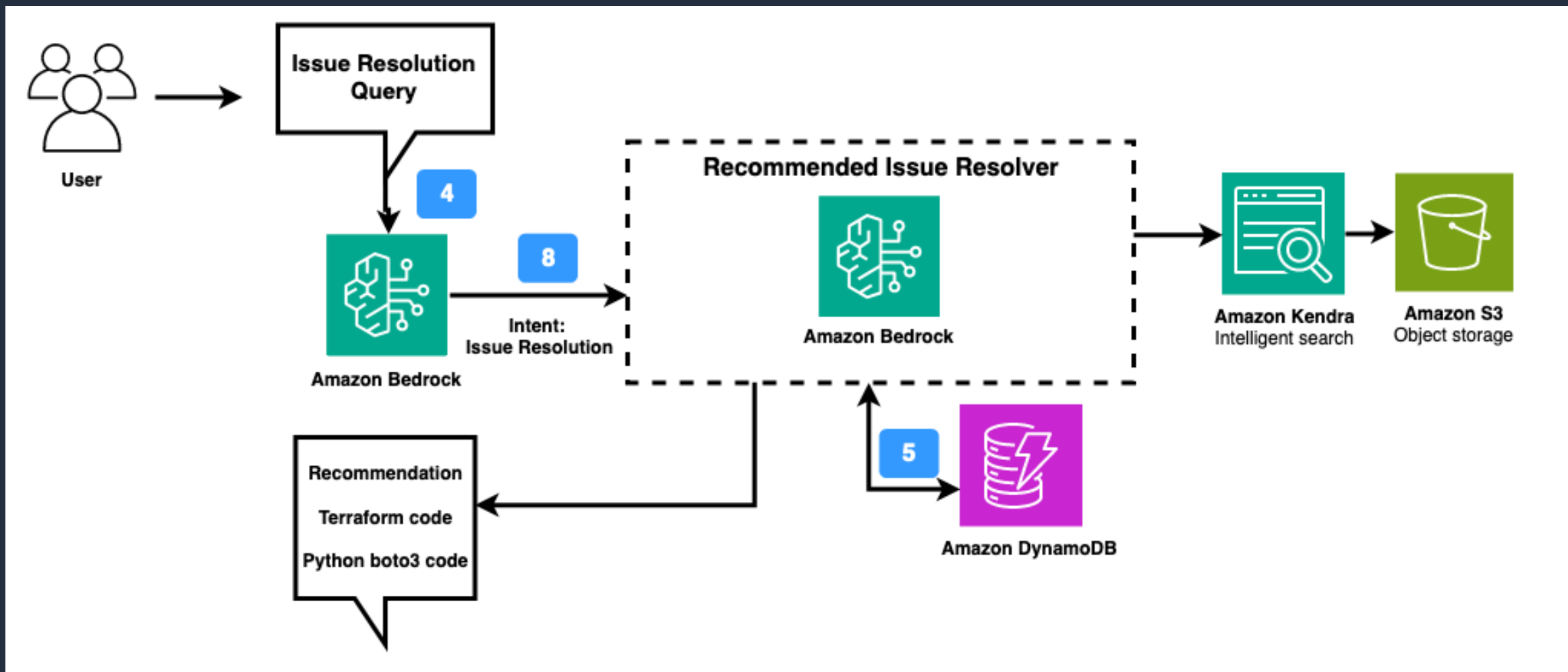
BMW GROUP ACCELERATES CLOUD OPTIMIZATION USING AMAZON BEDROCK



Health Check: get findings via API call



Issue Fixer: suggest remedies with RAG support



```
def query_issue_resolution_prompt(query):  
  
    context = Kendra().query_aws_docs(query)  
  
    template = """  
    Human:   
    You are an AWS Solutions Architect and expert in resolving  
    issues with AWS account and its services. You are asked to  
    respond to user queries with step by step recommendations  
    along with the terraform code which can be deployed in the  
    account.  
  
    .....  
  
    Context: {context}  
    """  
  
    response = Bedrock.invoke(template)  
    return response
```



please fix Low Utilization Amazon EC2 Instances

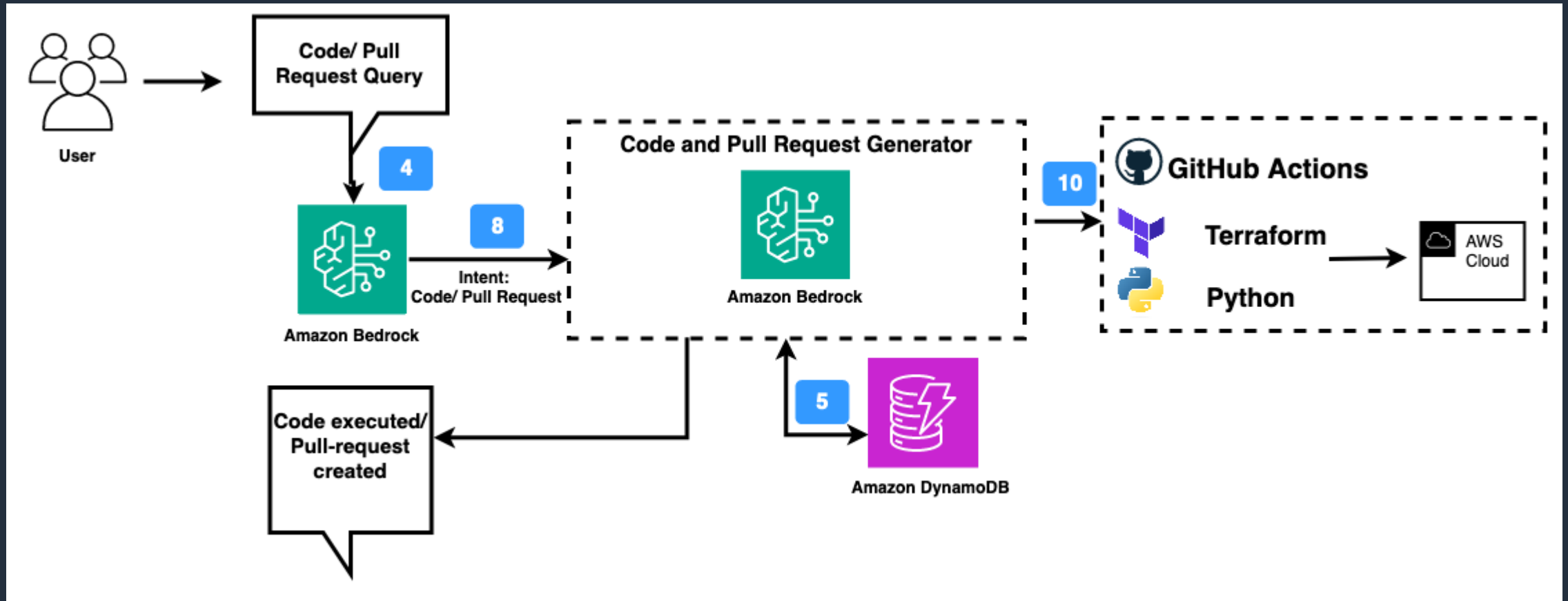
Here are the recommendations, Terraform code, and Python boto3 code to resolve the issue of low utilization of EC2 instances:

Recommendations:

- Use Auto Scaling groups to automatically scale in/out EC2 instances based on utilization metrics like CPU usage
- Resize underutilized instances to smaller instance types to optimize costs
- Schedule start/stop of dev/test instances to run only when needed

Query knowledge base for issue and create prompt for Bedrock model

Code and pull request generator



BMW GROUP ACCELERATES CLOUD OPTIMIZATION USING AMAZON BEDROCK

```
def template_for_deployment_queries():  
    response = ""  
    Human:  
    You are an AWS Solution architect, you are asked to fix Trusted Advisor warnings,  
    for which you receive recommendations in the chat history that contains terraform  
    and python code using boto3 package, generated previously.  
  
    .....  
  
    Previous conversation history:  
    {chat_history}  
    Instruction: {input}  
    Assistant:  
    {agent_scratchpad}  
    ""  
    return response
```

Build prompt (shortened)

```
def deployment_query_agent(query, sessionId):  
    message_history = SessionsTable.load_chat_message_history(sessionId)  
    memory = ConversationBufferWindowMemory(  
        ....  
    )  
    agent_executor_conv = initialize_agent(  
        ....  
    )  
  
    prompt_template = template_for_deployment_queries()  
    agent_executor_conv.agent.llm_chain.prompt.template = prompt_template  
  
    return get_llm_response(agent_executor_conv, query)
```

Query model with prompt (shortened)



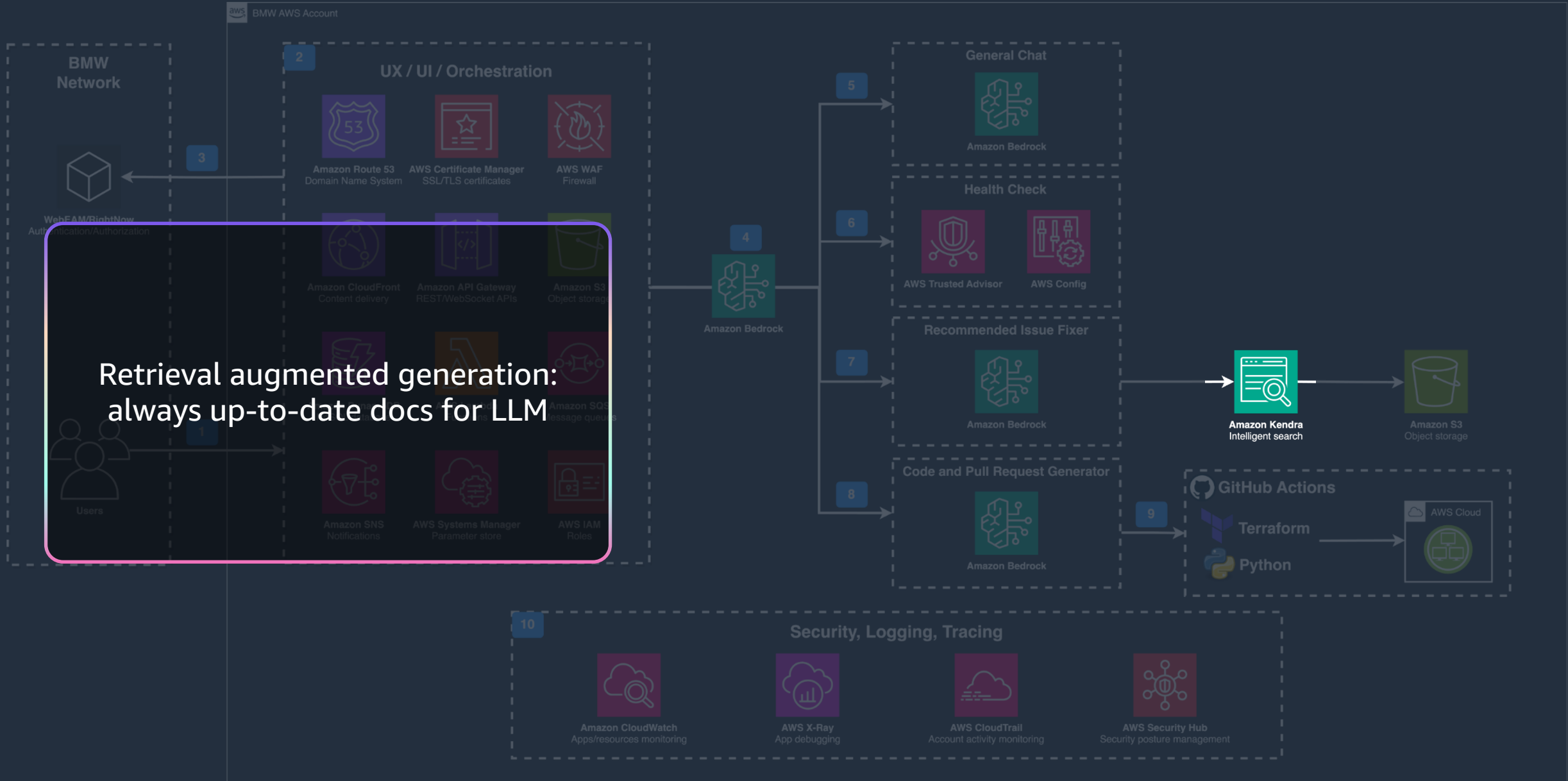
```
resource "aws_launch_configuration" "web" {  
    image_id      = data.aws_ami.ubuntu.id  
    instance_type = "t2.micro"  
  
    lifecycle {  
        create_before_destroy = true  
    }  
}  
  
resource "aws_autoscaling_group" "web" {  
    launch_configuration = aws_launch_configuration.web.name  
    min_size             = 1  
    max_size             = 2  
  
    target_group_arns = [aws_lb_target_group.asg.arn]  
  
    tag {  
        key           = "Name"  
        value         = "web"  
        propagate_at_launch = true  
    }  
}
```

Terraform code

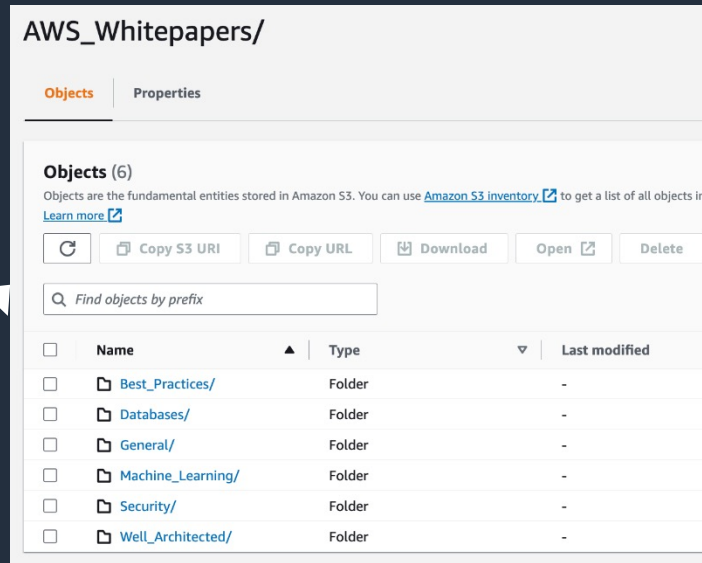
```
import boto3  
  
asg_client = boto3.client('autoscaling')  
ec2_client = boto3.client('ec2')  
  
def scale_asg(asg_name, new_size):  
    asg_client.update_auto_scaling_group(  
        AutoScalingGroupName=asg_name,  
        DesiredCapacity=new_size  
    )  
  
def get_asg_instances(asg_name):  
    asg = asg_client.describe_auto_scaling_groups(AutoScalingGroupNames=[asg_name])  
    instance_ids = [i['InstanceId'] for i in asg['AutoScalingGroups'][0]['Instances']]  
  
    instances = ec2_client.describe_instances(InstanceIds=instance_ids)  
  
    return instances  
  
# Usage:  
asg_name = 'my-asg'  
scale_asg(asg_name, 3)  
instances = get_asg_instances(asg_name)
```

Python code

BMW GROUP ACCELERATES CLOUD OPTIMIZATION USING AMAZON BEDROCK



Using Kendra on S3 as RAG – deep dive.



Source: AWS documentation
(downloaded automatically)

```
import boto3
client = boto3.client("kendra")
exemplary_query = "what is machine learning?"
response = client.retrieve(IndexId=<Index_id>, QueryText=exemplary_query)
```

Build query to search RAG

```
{
  "Query": "what is machine learning?",
  "DocumentTitle": "machine-learning-foundations.pdf",
  "Content": "Amazon Web Services Machine Learning Foundations ...",
  "DocumentURI": "https://kendra-awsdocs....s3.amazonaws.com/AWS_Whitepapers/ML/ML-foundations.pdf"
}
```

Return response (shortened)



BMW In-Console Cloud Assistant

How may I help you?

What is the health status of my AWS Account 94 [redacted] 944?

Send >>

BMW In-Console Cloud Assistant

Connected

Benefits

- Scalable solution
- Supports our DevOps teams
- Facilitates continuous learning
- Easy maintenance and extendibility



What's coming next?

- Link bot with FinOps-toolchain
- Accelerate development
- Analyze AWS services directly





Thank you!

Dr. Jens Kohl

Jens.Kohl@bmw.de

Christian Müller

cmr@amazon.de



Please complete the session survey.

