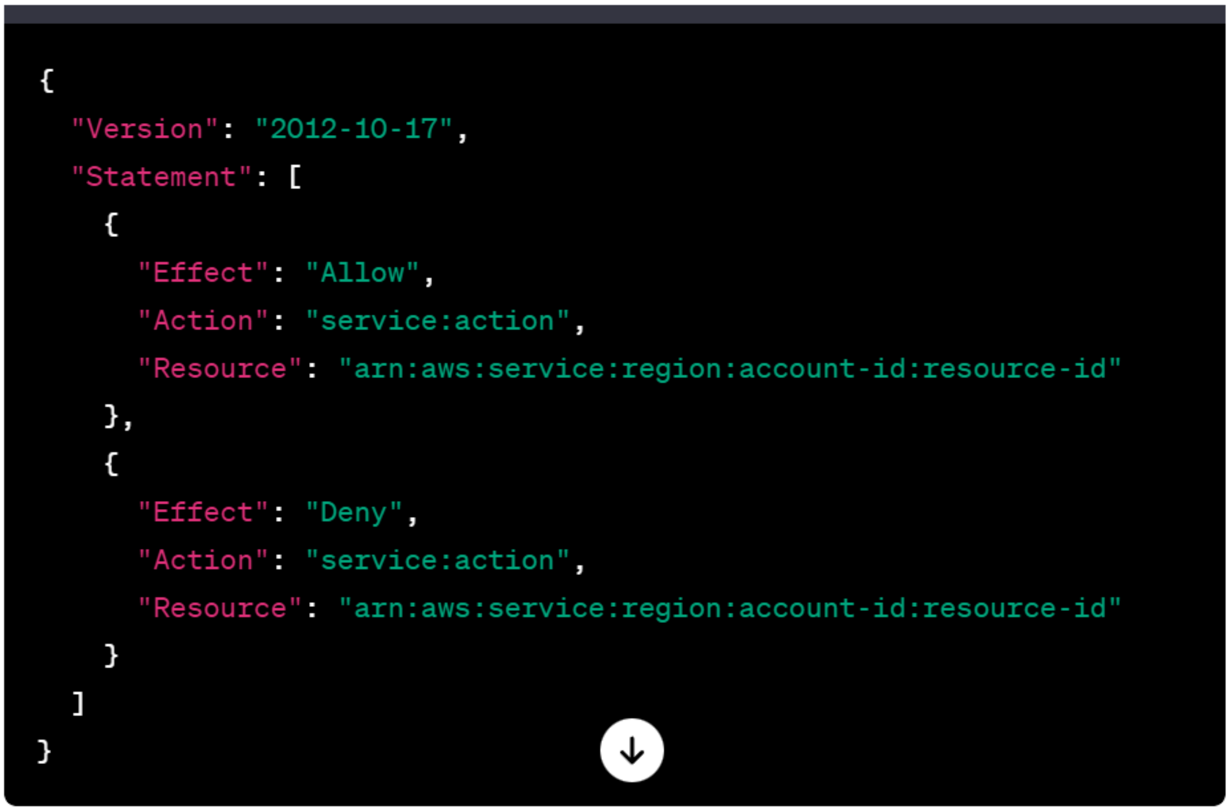
AWS Identity and Access Management (IAM) policies can be a bit complex due to the extensive range of services and actions available. Below is a cheatsheet to help you get started with AWS IAM policies. Please note that this is a general guide, and you should tailor the policies based on your specific requirements.

IAM Policy Structure:



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "service:action",
      "Resource": "arn:aws:service:region:account-id:resource-id"
    },
    {
      "Effect": "Deny",
      "Action": "service:action",
      "Resource": "arn:aws:service:region:account-id:resource-id"
    }
  ]
}
```

IAM Policy Structure:

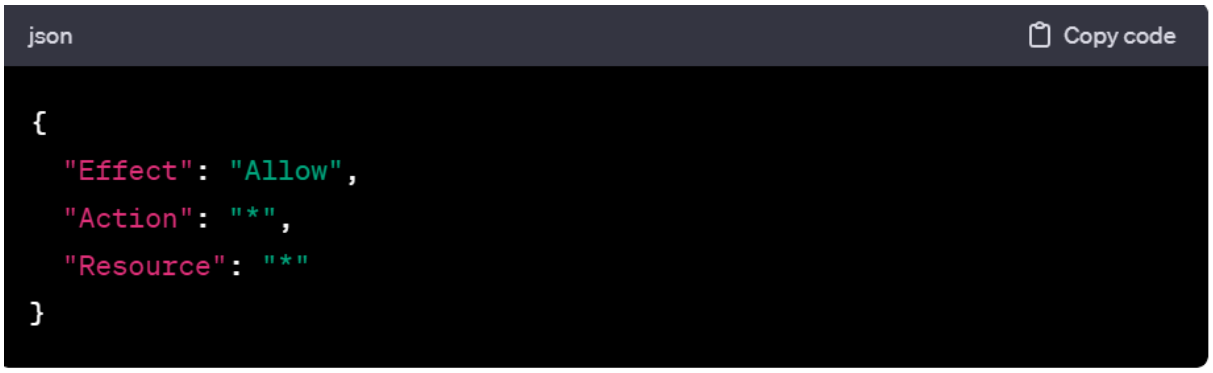
IAM policies have the following basic structure

Version: The policy language version, usually set to "2012-10-17."

- Statement: An array of statements, each with an "Effect," "Action," and "Resource.

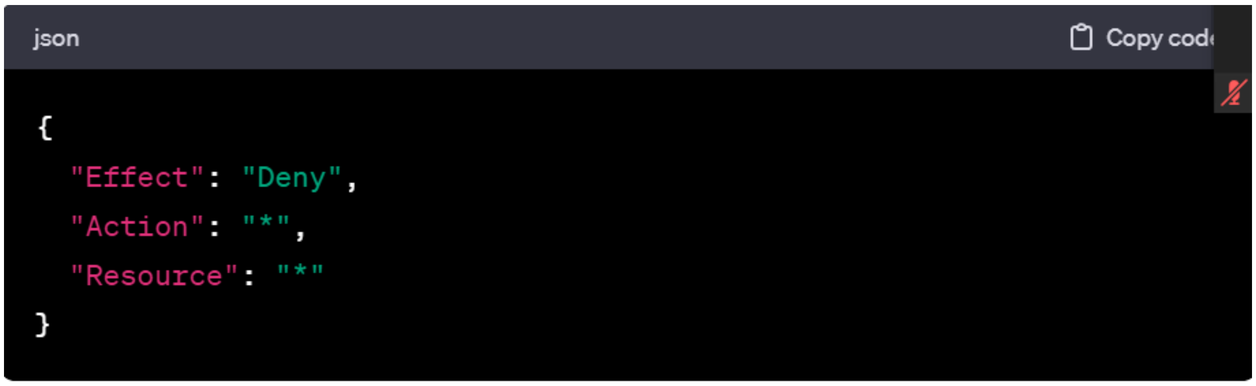
Basic Actions:

Allow All Actions on All Resources:



```json
{
  "Effect": "Allow",
  "Action": "*",
  "Resource": "*"
}
```

Deny All Actions:



```json
{
  "Effect": "Deny",
  "Action": "*",
  "Resource": "*"
}
```

- Version: The policy language version, usually set to "2012-10-17."
- Statement: An array of statements, each with an "Effect," "Action," and "Resource."

Basic Actions:

Allow All Actions on All Resources:

json

Copy code

```
 "Effect"  "Allow"
 "Action"  "*"
 "Resource"  "*"
```

Deny All Actions:

json

Copy code

```
 "Effect"  "Deny"
 "Action"  "*"
 "Resource"  "*"
```

Permissions on Specific Services:

Allow All Actions on a Specific Service:

json

Copy code

```
 "Effect"  "Allow"
 "Action"  "service:*"
 "Resource"  "*"
```

Allow Specific Action on a Specific Resource:

json

Copy code

```
 "Effect"  "Allow"
 "Action"  "service:action"
 "Resource"  "arn:aws:service:region:account-id:resource-id"
```

Conditional Statements:

Allow Action if a Condition is Met:

json

Copy code

```
"Effect"  "Allow"
"Action"  "service:action"
"Resource"  "arn:aws:service:region:account-id:resource-id"
"Condition"
"ConditionKey"  "service:condition-key"
"StringValueEquals"
"service:condition-value"  "desired-value"
```

Resource-Level Permissions:

Allow Actions on Resources with a Specific Tag:

json

Copy code

```
"Effect"  "Allow"
"Action"  "service:action"
"Resource"  "arn:aws:service:region:account-id:resource-type/resource-id"
"Condition"
"StringEquals"
"aws:RequestTag/key"  "value"
"aws:ResourceTag/key"  "value"
```

Deny Statements:

Deny Specific Action:

json

Copy code

```
"Effect"  "Deny"
"Action"  "service:action"
"Resource"  "arn:aws:service:region:account-id:resource-id"
```

Example Policy for EC2 Full Access:

json

Copy code

```
"Version"  "2012-10-17"
"Statement"

"Effect"  "Allow"
"Action"  "ec2:*"
"Resource"  "*"
```

Best Practices:

Principle of Least Privilege (PoLP): Only grant the permissions necessary for a user or role to perform their job.
Use Groups: Assign policies to IAM groups and add users to groups based on their roles.
Regular Review: Periodically review and update policies to ensure they align with current requirements.
Avoid Wildcards: Be specific in your actions and resource definitions to reduce the risk of unintended permissions.
Use IAM Roles for EC2 Instances: Instead of using access keys, assign IAM roles to EC2 instances to grant them permissions.

Remember to replace placeholders like "service," "action," "region," "account-id," and "resource-id" with your specific values.