Manchester Metropolitan University

CLOUD COMPUTING

/PLATFORM as a SERVICE (PaaS)
/AWS ELASTIC BEANSTALK

Dr Mohammed Kaleem

m.kaleem@mmu.ac.uk

CONTENTS

/01 UNDERSTANDING THE DIFFERRENCE BETWEEN LAMBDA AND ELASTIC BEANSTALK

/02 DEPLOYING JAVA APPS TO AWS BEANSTALK

- How to deploy a java app to AWS elastic beanstalk
  - Creating a service role
  - Configuring AWS elastic beanstalk and deploying a WAR file application

AWS LAMBDA Vs AWS ELASTIC BEANSTALK

Both Lambda and Elastic Beanstalk are popular services offered by Amazon Web Services (AWS) for deploying and managing applications, but they differ significantly in their architecture and use cases.

Elastic Beanstalk

Lambda

AWS LAMBDA

AWS Lambda is a serverless compute service that allows you to run code without provisioning or managing servers.

Key Features:

- **Event-driven**: Executes code in **response to events** such as changes to data in Amazon S3, DynamoDB, or invoking HTTP endpoints via API Gateway.
- **Pay-per-use pricing**: Billed based on the number of requests and the duration of code execution, with no charge when the code is not running. **2m free request.**
- **Supports multiple programming languages** including Node.js, Python, Java, and more.
- Good Use Cases:
  - Real-time file processing
  - IoT data processing
  - Backend for mobile or web applications
  - Chatbots and voice-enabled applications

AWS ELASTIC BEANSTALK

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with popular programming languages, frameworks, and containers.

Key Features:

- **Fully managed platform:** AWS handles infrastructure provisioning, deployment, and scaling, **allowing developers to focus on application development**.
- Support for multiple platforms: Offers support for **Java**, .NET, Node.js, Python, Ruby, Go, Docker, and more.
- **Customizable environment configurations:** Allows customization of underlying resources such as EC2 instance types, auto-scaling configurations, and load balancers.
- Use Cases:
  - **Web application hosting**
  - API services
  - Microservices architectures
  - Development and testing environments

AWS LAMBDA Vs AWS ELASTIC BEANSTALK

Deployment Model:

- AWS Lambda: Serverless, event-driven model.
- AWS Elastic Beanstalk: Managed platform-as-a-service (PaaS) model.

Scaling:

- AWS Lambda: Scales automatically based on the number of incoming requests.
- AWS Elastic Beanstalk: Offers auto-scaling capabilities based on defined metrics such as CPU utilization or request count.

Management Complexity:

- AWS Lambda: Minimal management required as AWS handles infrastructure provisioning and scaling.
- AWS Elastic Beanstalk: Requires more configuration and management compared to Lambda, but less than managing raw infrastructure.

Pricing:

- AWS Lambda: Pay-per-use pricing based on the number of requests and duration of execution.
- AWS Elastic Beanstalk: Pay for underlying resources such as EC2 instances, load balancers, and storage, with additional charges for data transfer and other services.

TYPICAL USE CASES

AWS Lambda:

Ideal for event-driven applications with short-lived, stateless functions.

Suitable for applications with unpredictable or sporadic workloads.

AWS Elastic Beanstalk:

Best suited for long-running applications requiring consistent performance.

Provides more control over underlying infrastructure and configuration.

Choosing between Lambda and Elastic Beanstalk depends on factors such as application architecture, expected workload, and management preferences.

ALL DEPENDS ON USE CASE

- AWS Lambda and AWS Elastic Beanstalk **are both powerful services** offered by AWS for deploying and managing applications.
- Choose AWS **Lambda for event-driven**, serverless architectures with **short-lived functions** and unpredictable workloads.
- Opt for AWS **Elastic Beanstalk when deploying long-running applications** that require more control over underlying infrastructure and configuration.
- Consider your specific use case, workload requirements, and management preferences when selecting between Lambda and Elastic Beanstalk.

DEPLOYING AN APP TO AWS ELASTIC BEANSTALK

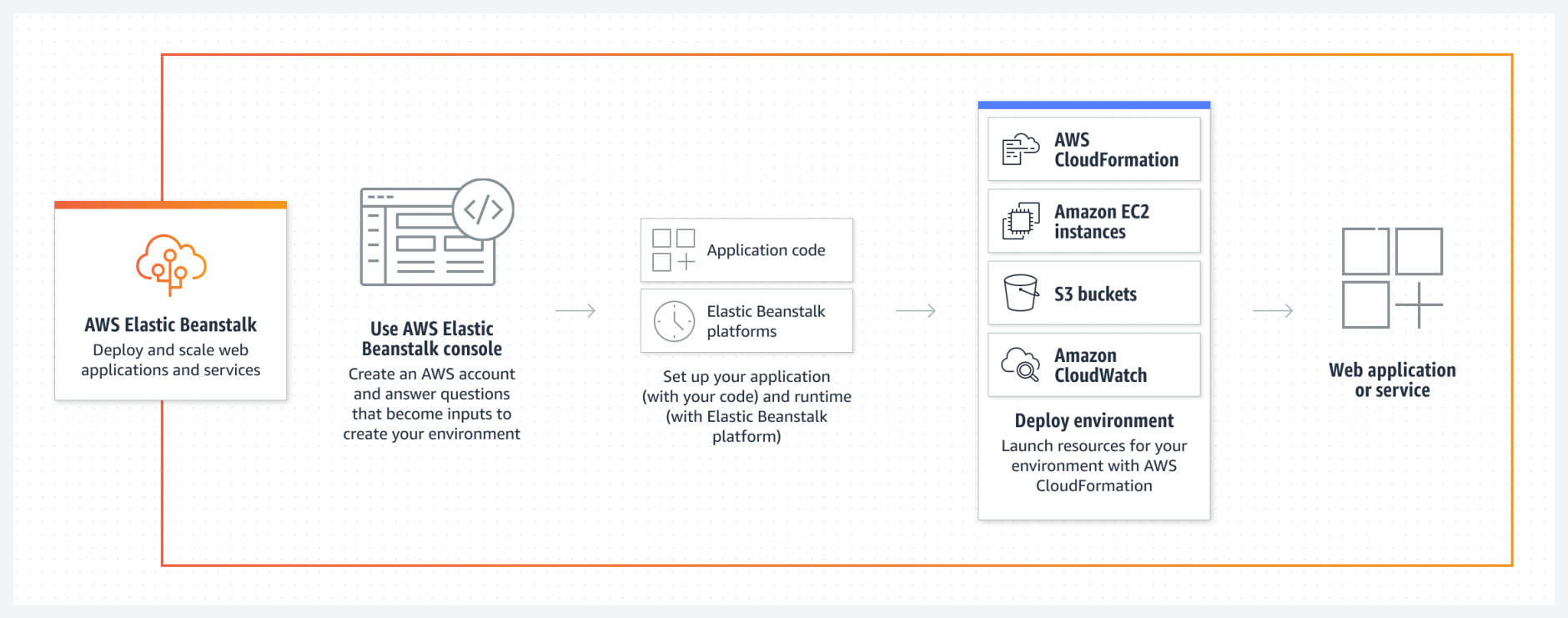
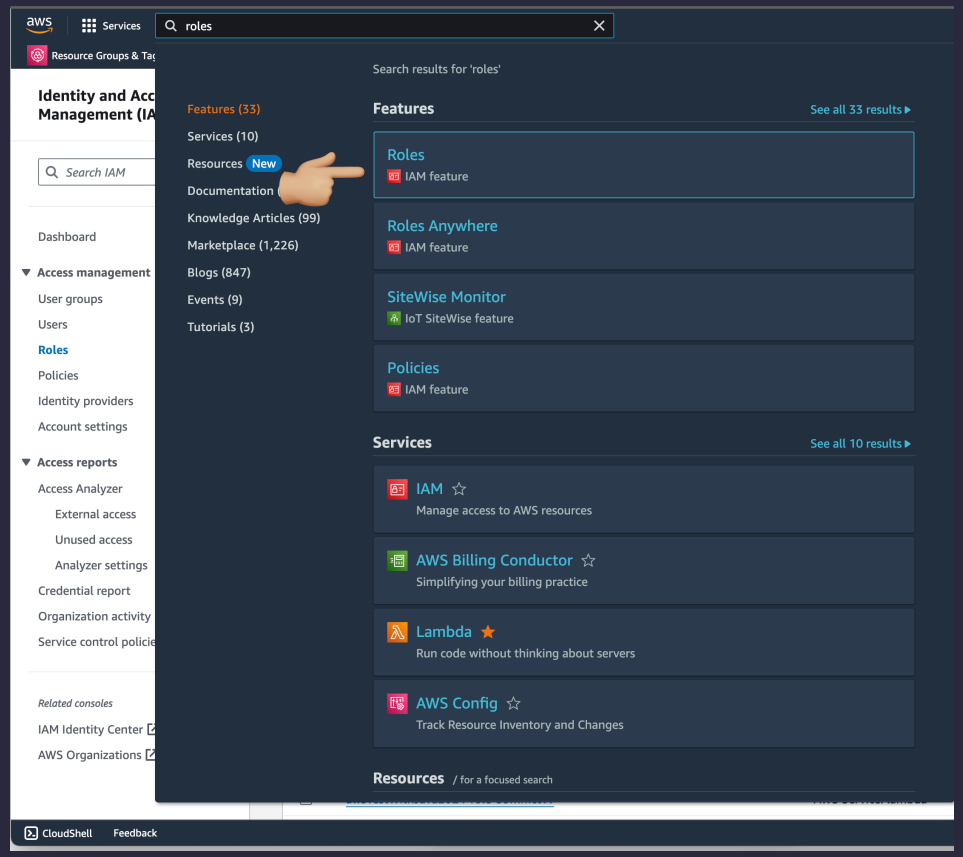


image credit: https://aws.amazon.com/elasticbeanstalk/

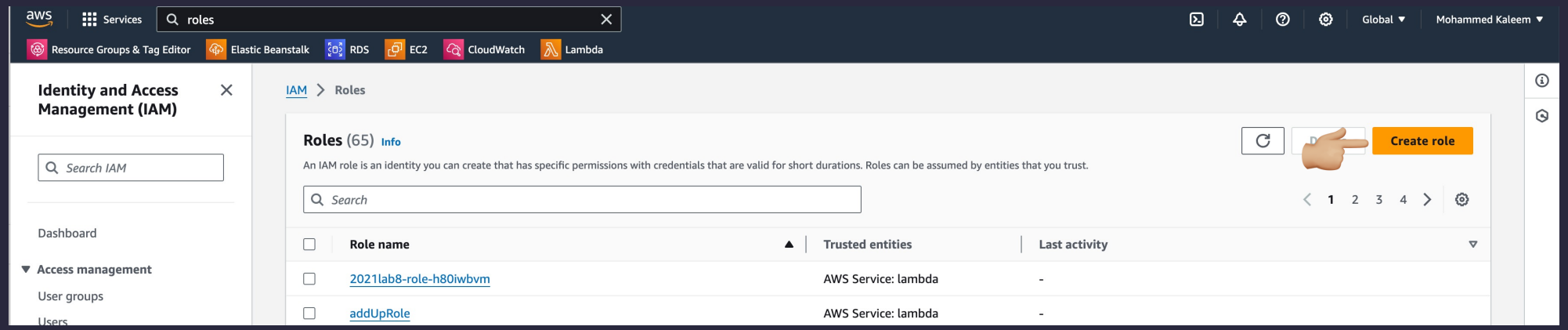
STEP 1: CREATING A SERVICE ROLE

- A service role is the AWS Identity and Access Management role (**IAM role**) that Elastic Beanstalk assumes when calling other services on your behalf.
- For example, Elastic Beanstalk uses a service role when it calls Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing, and Amazon EC2 Auto Scaling APIs to gather information.
- The service role that Elastic Beanstalk uses is the one that you specified when you create the Elastic Beanstalk environment.
- AWS Elastic Beanstalk takes away all EC2 VM configuration and does all that for you using the service role.

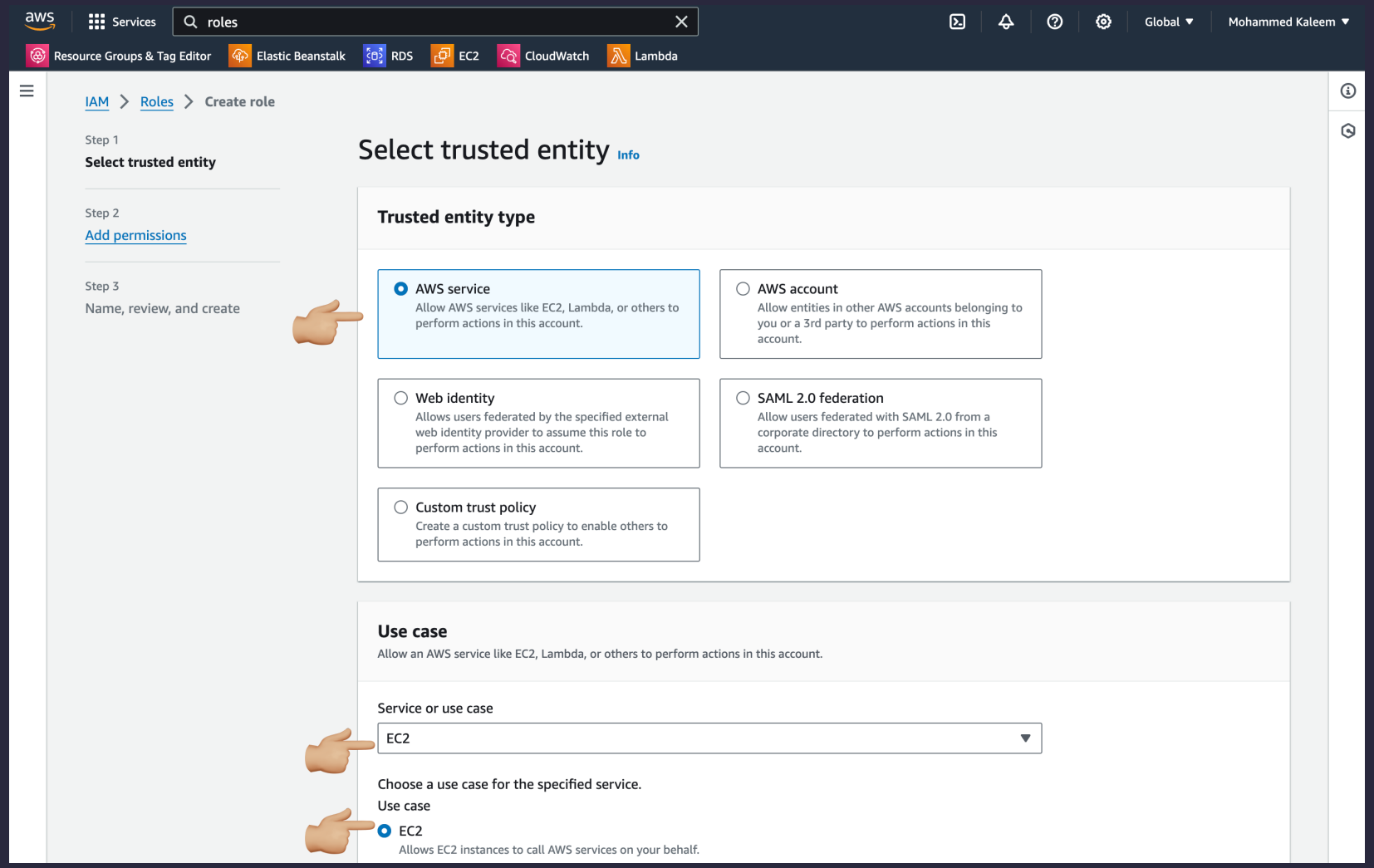
CREATING A SERVICE ROLE



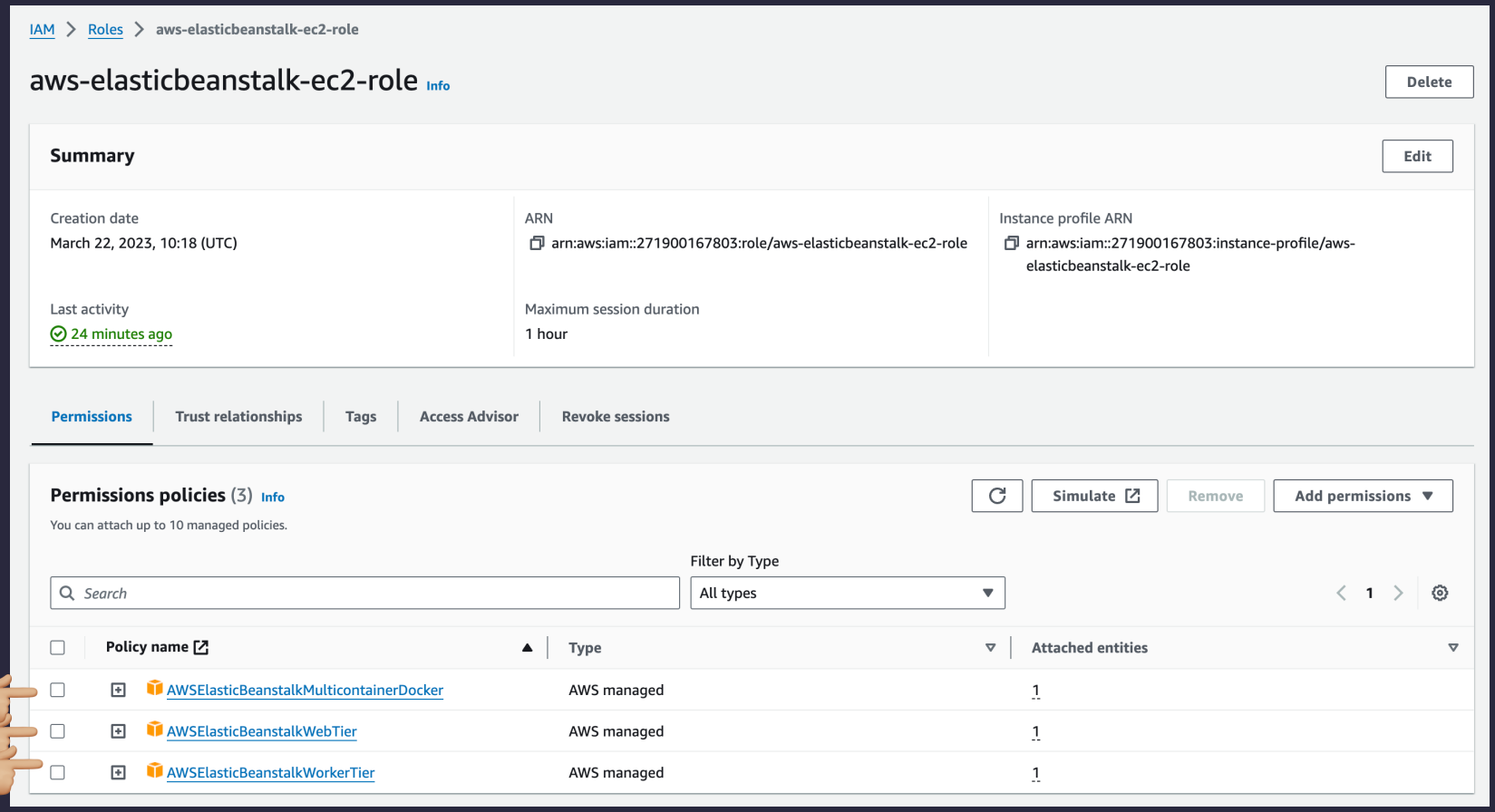
CREATING A SERVICE ROLE



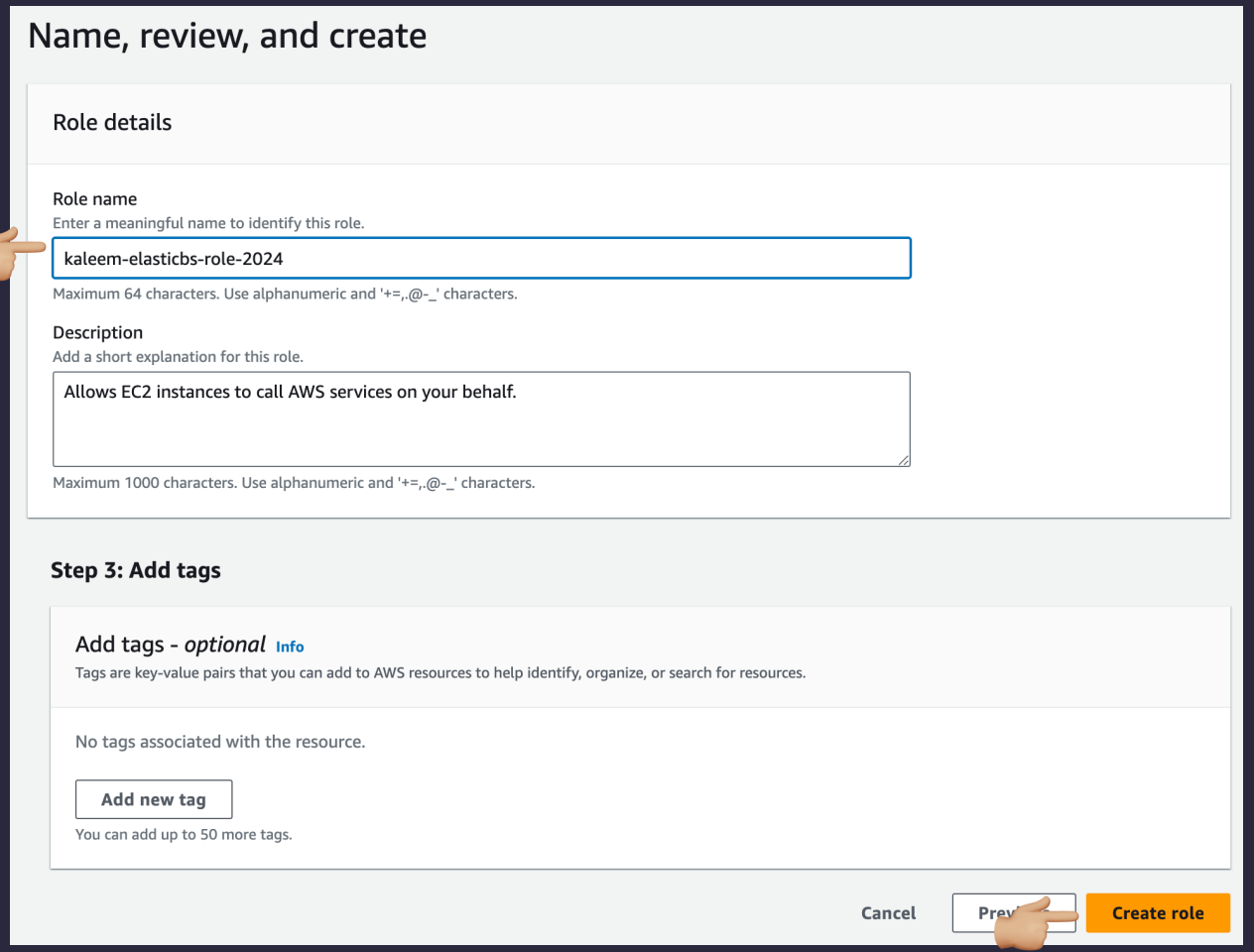
CREATING A SERVICE ROLE



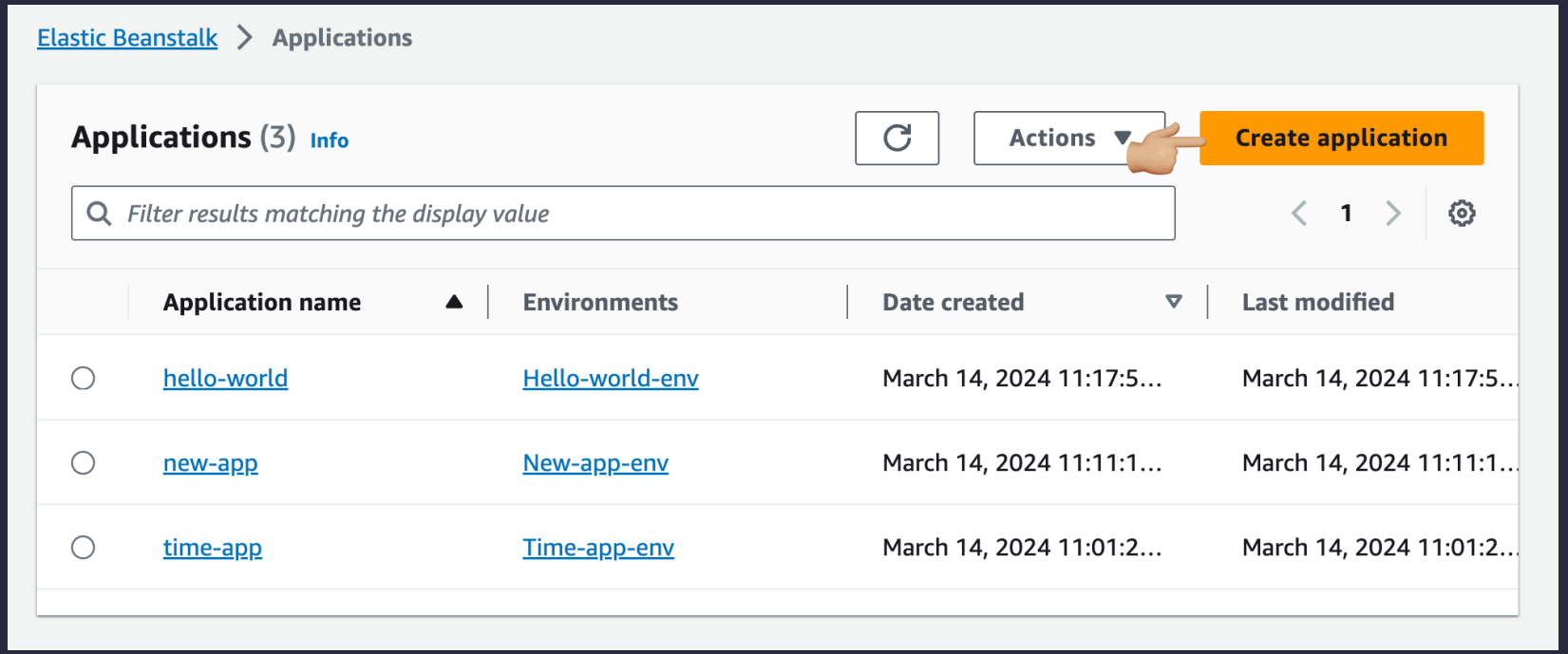
CREATING A SERVICE ROLE - add permissions



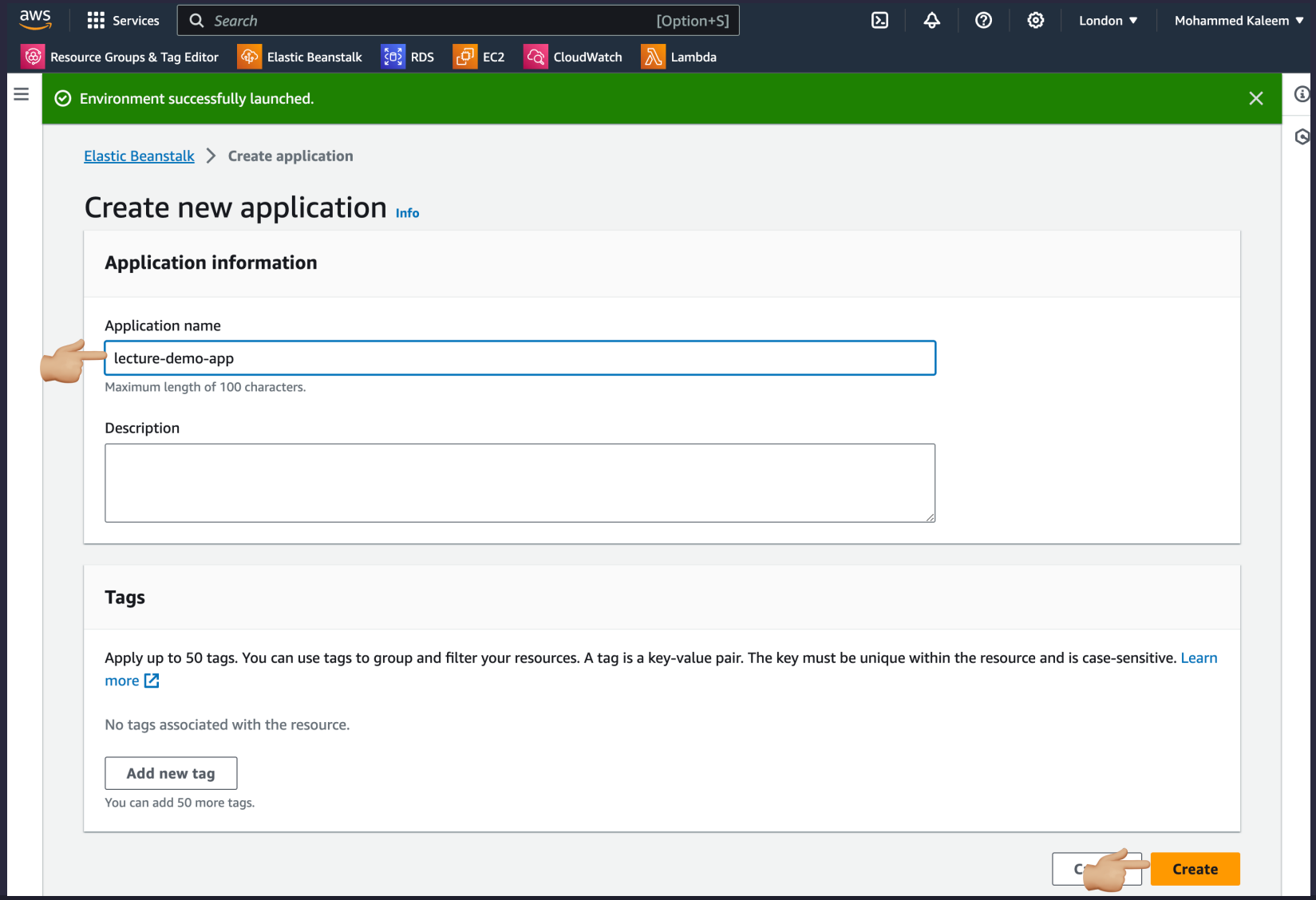
CREATING A SERVICE ROLE – name and save



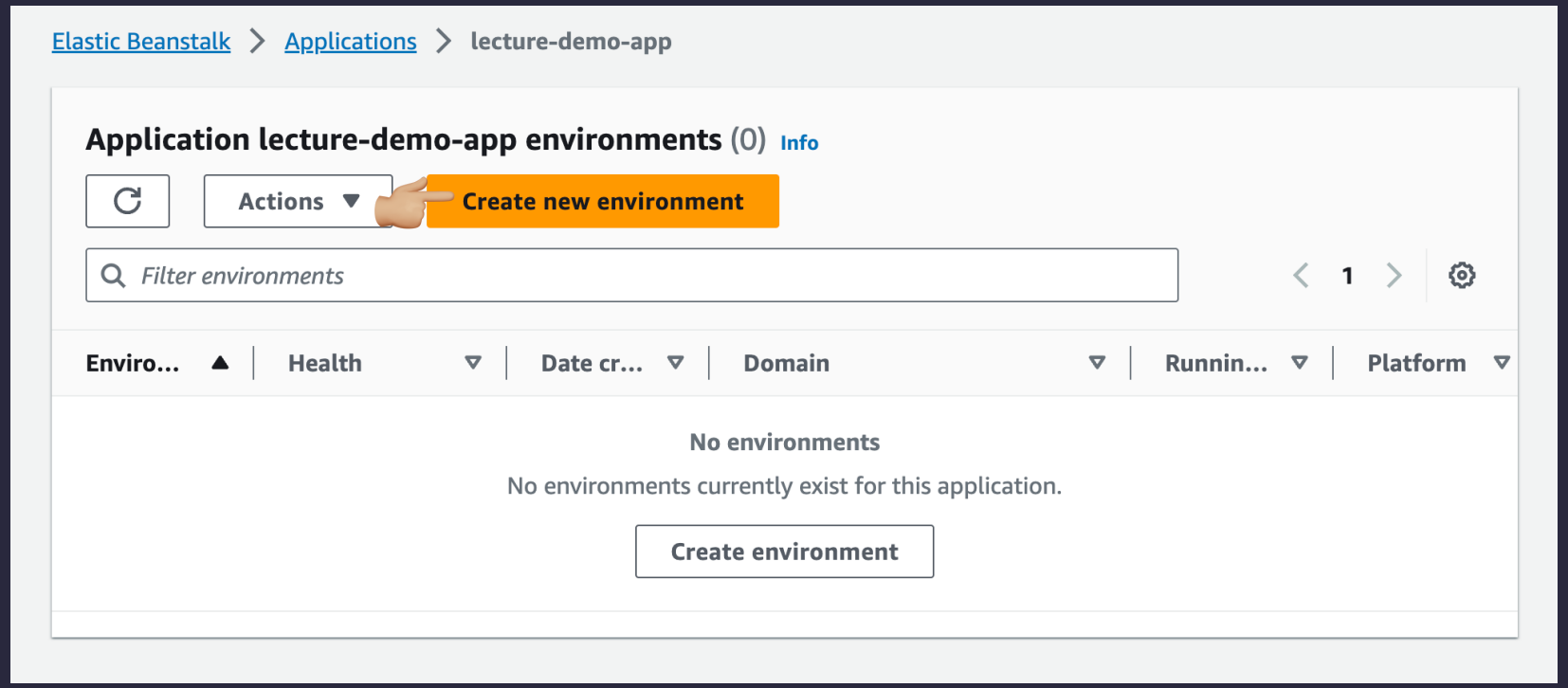
STEP 2: CREATING A BEANSTALK APP



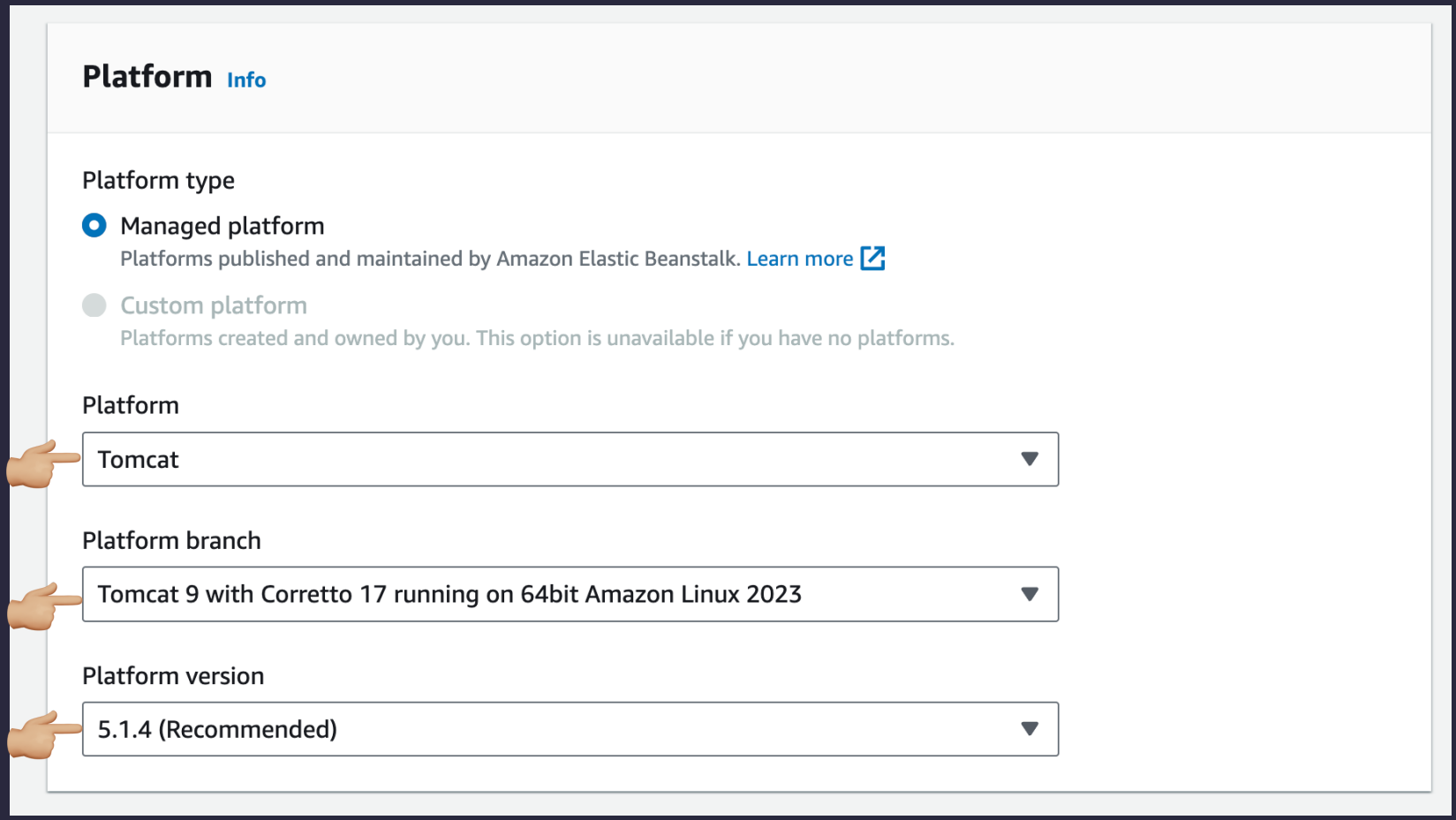
CREATING A BEANSTALK APP – name your application



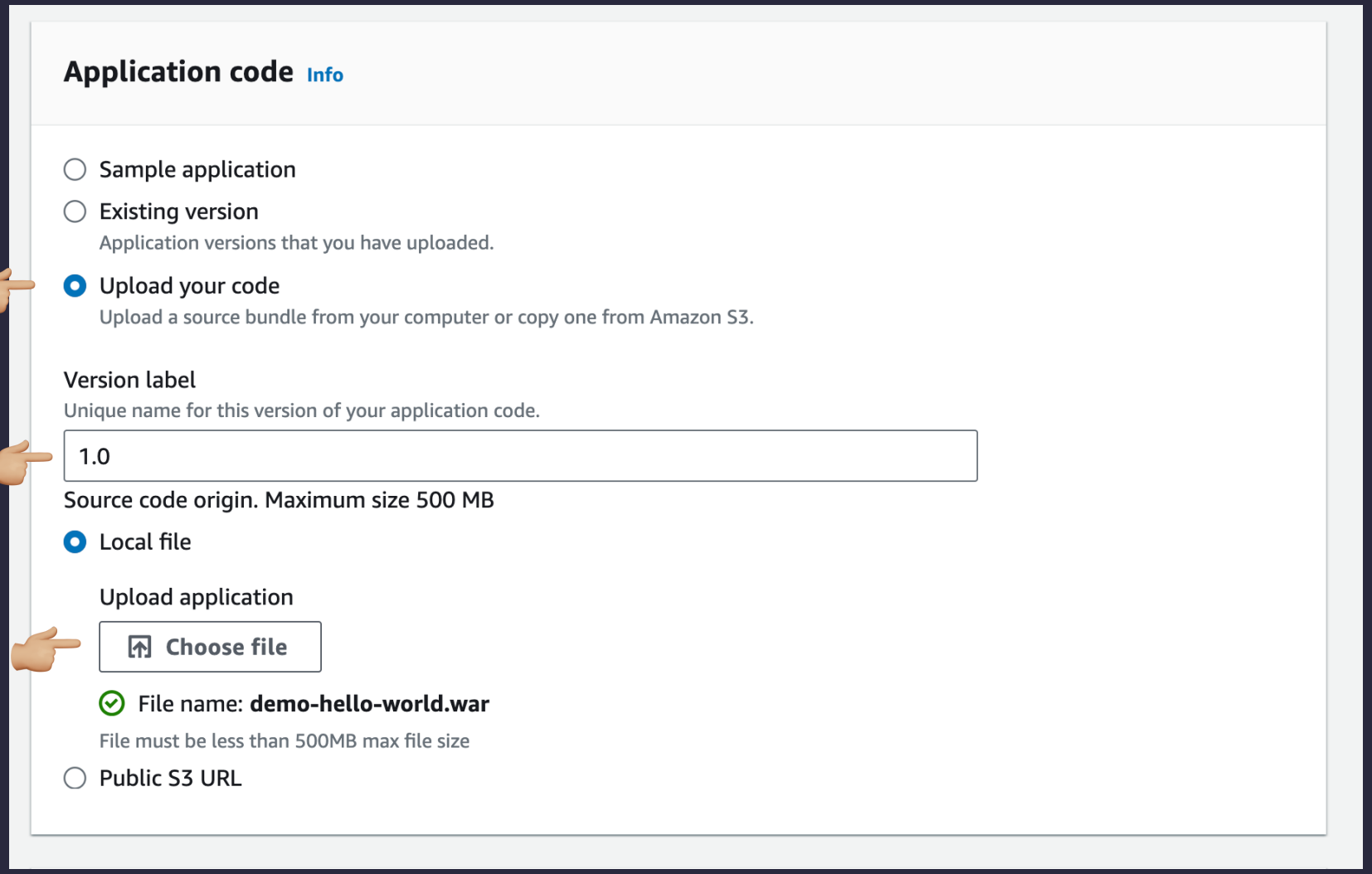
CREATING A BEANSTALK APP – create environment



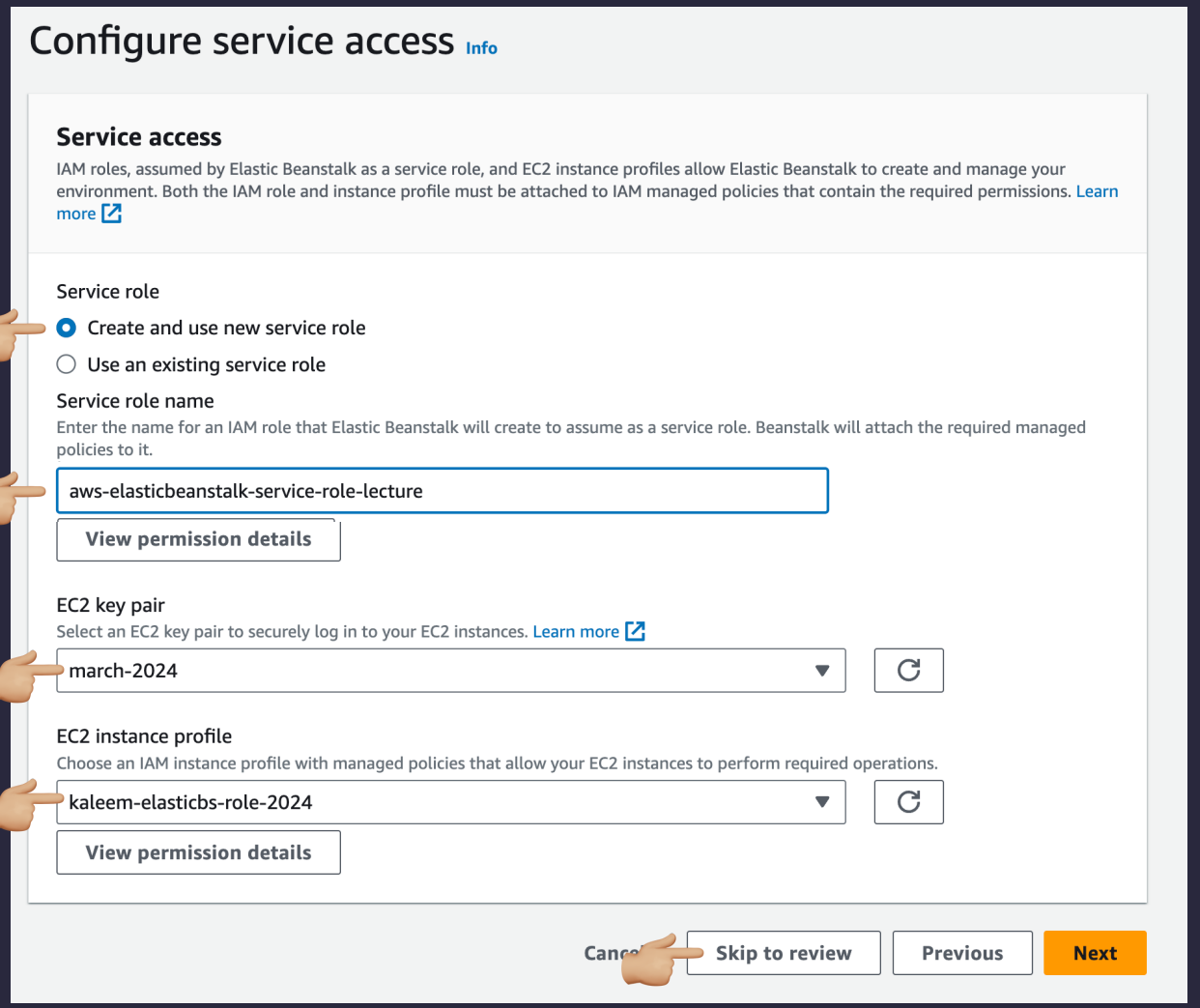
CREATING A BEANSTALK APP – select your platform



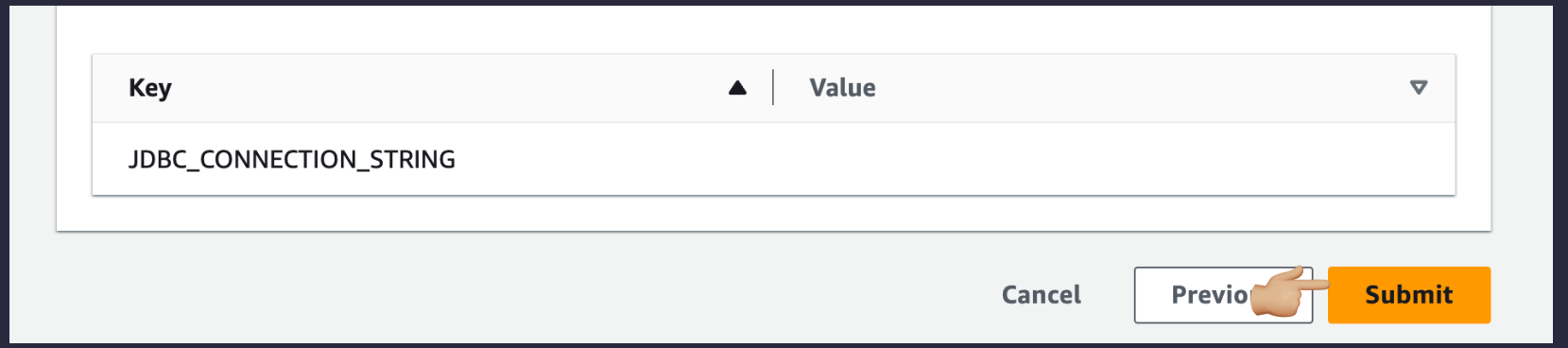
CREATING A BEANSTALK APP – upload code



CREATING A BEANSTALK APP

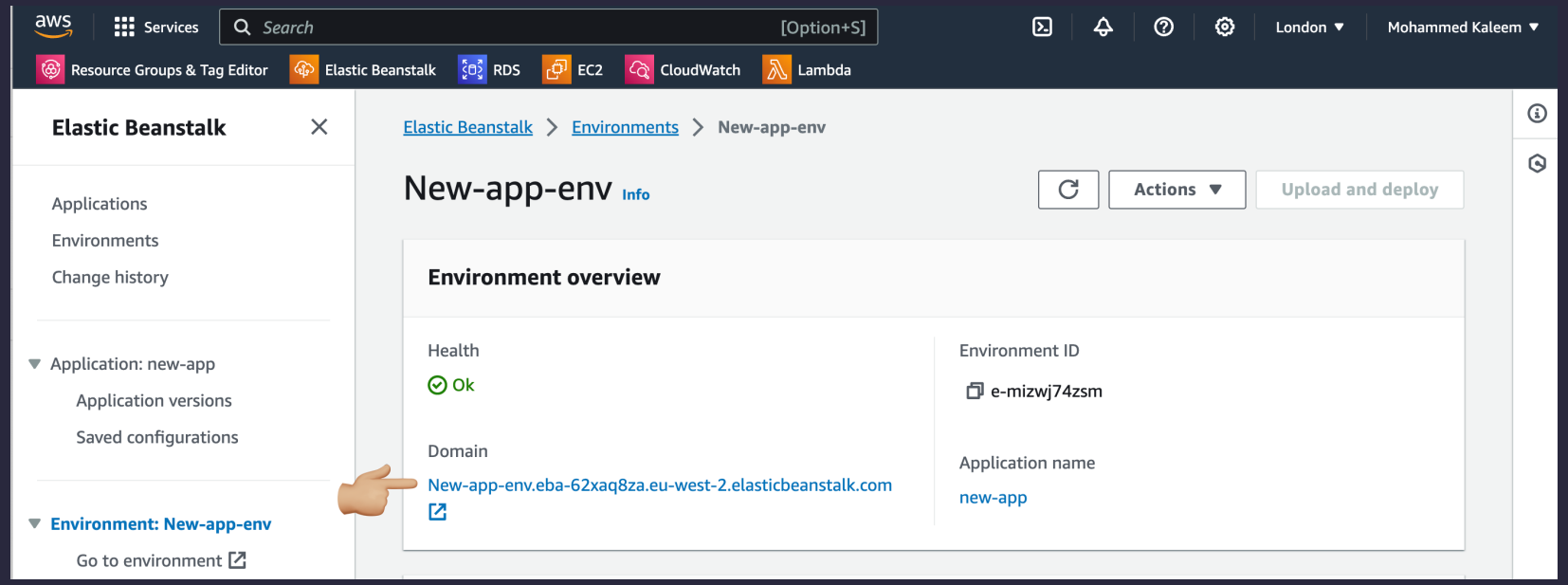


CREATING A BEANSTALK APP – review and submit



⏳ Then wait...

CREATING A BEANSTALK APP – wait for the URL



A few minutes later you will get the URL to your web application.

Summary

- Create the IAM Role
- Export application from eclipse as WAR or JAR depending on how you implemented the application.
  - Dynamic Web Applications should be packaged as WAR files.
  - Spring boot applications should be packaged as JAR files.
- Configure new elastic beanstalk application/environment.
- Wait a while…
- App will be deployed, all set up and deployment handled by AWS, you'll get a URL to your app at the end of the process.