Desenvolvimento para Servidores-II

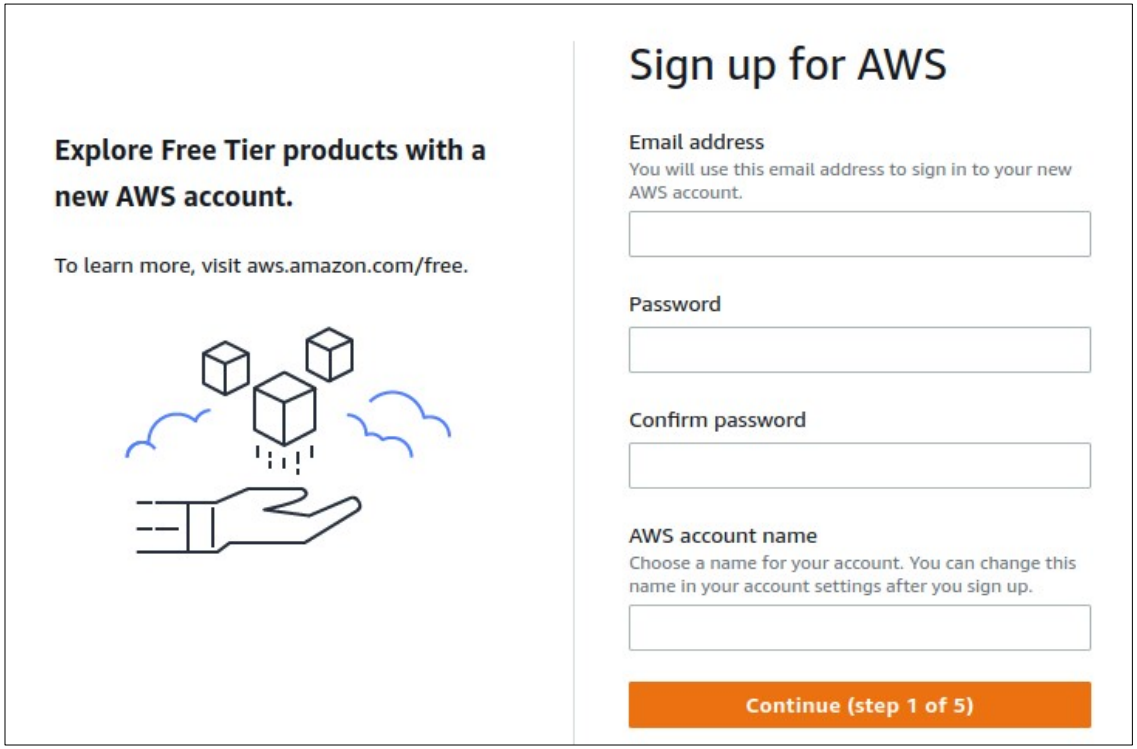
Upload de Arquivos

Neste tópico abortaremos a criação de um *end point* na API REST para fazer *upload* de arquivos num *bucket* do Amazon S3 e localmente.

Prof. Ciro Cirne Trindade

Criar uma conta na AWS

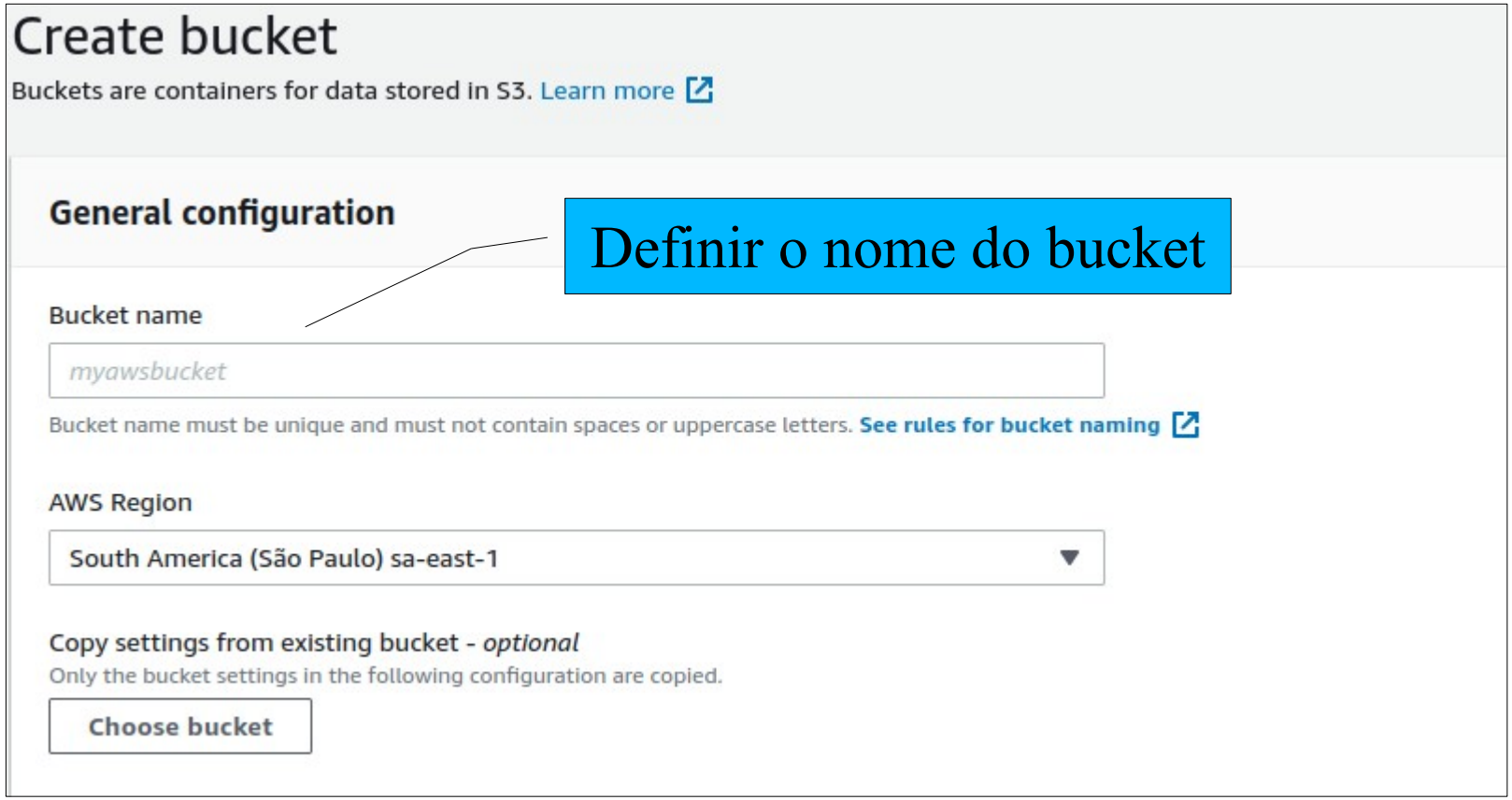
- Acessar aws.amazon.com
- Clicar no botão **Get Started for Free**



Amazon Simple Storage Service (S3)

- S3 é um serviço de armazenamento em nuvem da AWS
- Os dados são armazenados como objetos em recursos chamados buckets
- Para criar um bucket, logue no console da AWS, selecione S3 e clique no botão

Amazon Simple Storage Service (S3)



Amazon Simple Storage Service (S3)

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. **Learn more**

- [ ] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [ ] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [ ] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [ ] **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [ ] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Desmarcar

Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

- [x] I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Marcar

Create bucket

Amazon Simple Storage Service (S3)

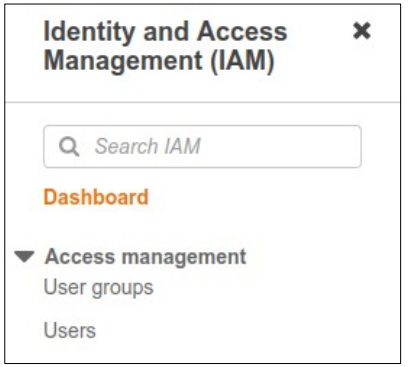
- Tornando o bucket de acesso público para leitura
  - Em Amazon S3, clique no bucket que foi criado
  - Clique na aba Permissões
  - Edite a Política do bucket e copie o arquivo JSON do próximo slide

Amazon Simple Storage Service (S3)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPublicRead",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::nome-do-bucket/*"
        }
    ]
}
```

Amazon Simple Storage Service (S3)

- Configurando o Identity and Access Management (IAM)
  - Acesse o IAM
  - Crie um grupo como política de acesso AmazonS3FullAccess
  - Crie um usuário com tipo de acesso programático
  - Adicione o usuário ao grupo criado
  - Baixe o arquivo CSV com o id e chave de acesso do usuário



Configurações o acesso ao bucket

- No arquivo application.properties inclua as propriedade de acesso ao bucket do S3

```
aws.access_key_id=ID-DA-CHAVE-DE-ACESSO
aws.secret_access_key=CHAVE-SECRETA
s3.bucket=NOME-DO-BUCKET
s3.region=sa-east-1
```

Dependência do AWS e Commons-io

- Dependência no pom.xml

```
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk</artifactId>
    <version>LATEST</version>
</dependency>

<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>LATEST</version>
</dependency>
```

Criar uma classe de configuração do S3

```
@Configuration
public class S3Config {

   @Value("${aws.access_key_id}")
   private String accessKey;
   @Value("${aws.secret_access_key}")
   private String secretKey;
   @Value("${s3.region}")
   private String region;

   @Bean
   public AmazonS3 s3client() {
      BasicAWSCredentials awsCred =
           new BasicAWSCredentials(accessKey, secretKey);
      AmazonS3 s3client =  AmazonS3ClientBuilder.standard()
          .withRegion(Regions.fromName(region))
          .withCredentials(new AWSStaticCredentialsProvider(awsCred))
          .build();
      return s3client;
   }
}
```

Criar uma classe de serviço do S3 (1/2)

```
@Service
public class S3Service {
    @Autowired
    private AmazonS3 s3client;

    @Value("${s3.bucket}")
    private String bucketName;

    public URI upload(MultipartFile multipartFile) {
        try {
            String filename =
                    multipartFile.getOriginalFilename();
            InputStream input = multipartFile.getInputStream();
            String contentType = multipartFile.getContentType();
            return upload(input, filename, contentType);
        } catch (IOException e) {
            throw new RuntimeException("Erro de IO: " +
                                    e.getMessage());
        }
    }
```

java.net.URI

Criar uma classe de serviço do S3 (2/2)

```
    public URI upload(InputStream input, String
                      filename, String contentType) {
        try {
            ObjectMetadata metadata = new ObjectMetadata();

            metadata.setContentType(contentType);

            s3client.putObject(bucketName, filename,
                               input, metadata);
            return s3client
                    .getUrl(bucketName, filename)
                    .toURI();
        } catch (URISyntaxException e) {
            throw new RuntimeException(
                      "Erro ao converter URL para URI");
        }
    }
}
```

Criar um *end point* para fazer o upload

```
@RestController
@RequestMapping("/upload")
public class UploadController {
   @Autowired
   private S3Service service;

   @PostMapping("/s3")
   public ResponseEntity<Void> uploadFile(
           @RequestParam(name = "file") MultipartFile file) {
      URI uri = service.upload(file);
      return ResponseEntity.created(uri).build();
   }
}
```

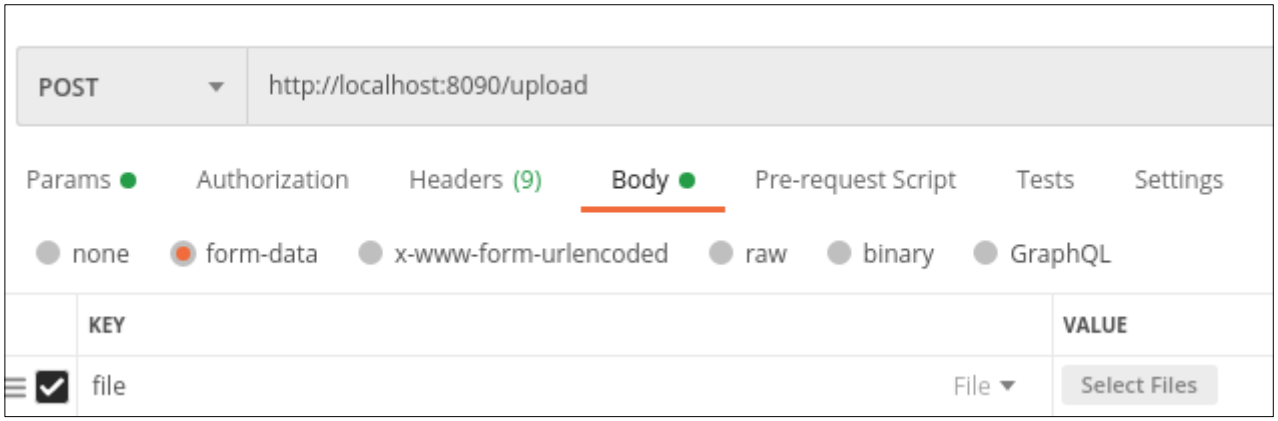
Configurando o tamanho máximo dos arquivos

- Para configurar o tamanho máximo dos arquivos que você pode fazer upload, coloque as instruções abaixo no arquivo application.properties

```
spring.servlet.multipart.max-file-size=1MB
spring.servlet.multipart.max-request-size=1MB
```

Testando no Postman

- Selecione o método POST, depois Body, form-data, defina o nome do parâmetro e seu tipo para file



Salvando o arquivo numa pasta local

- Também é possível salvar o arquivo em uma pasta local usando a biblioteca commons-io

UploadService.java (1/2)

```
@Service
public class UploadService {
    private final Path rootLocation;
    private static final String location = "uploadDir";

    public UploadService() throws IOException {
        rootLocation = Paths.get(location);
        try {
            if (!Files.exists(rootLocation)) {
                Files.createDirectories(rootLocation);
            }
        }
        catch (IOException e) {
             throw new IOException("Não foi possível criar o
diretório " + location);
        }
    }
```

Pasta onde os arquivos serão armazenados. Também pode ser um caminho absoluto

Se a pasta não existir, cria

UploadService.java (2/2)

```
public URI storeFile(MultipartFile arquivo) throws IOException,
                                                URISyntaxException {
    try {
        if (arquivo.isEmpty()) {
            throw new IOException("Falha: o arquivo está vazio.");
        }
        Path destinationFile = this.rootLocation
            .resolve(Paths.get(arquivo.getOriginalFilename()))
            .normalize().toAbsolutePath();
        if (!destinationFile.getParent().equals(
                    this.rootLocation.toAbsolutePath())) {
            throw new IOException("Não é possível guardar o arquivo
fora do diretório atual.");
        }
        try (InputStream inputStream = arquivo.getInputStream()) {
            Files.copy(inputStream, destinationFile,
                        StandardCopyOption.REPLACE_EXISTING);
            return new URI(destinationFile.toString());
        }
    } catch (IOException e) {
        throw new IOException("Falha ao guardar o arquivo.", e);
    }
}
}
```

Acrescenta o nome do arquivo à pasta onde ele será salvo

Copia o arquivo para a pasta

UploadController.java

```
@RestController
@RequestMapping("/upload")
public class UploadController {
   @Autowired
   private S3Service service;

   @Autowired
   private UploadService service;

   ...

   @PostMapping(value = "/local")
   @PreAuthorize("hasAnyRole('ADMIN')")
   public ResponseEntity<Void> upload(@RequestParam("arquivo")
                                      MultipartFile arquivo) {
      try {
         URI uri = service.storeFile(arquivo);
         return ResponseEntity.created(uri).build();
      } catch (IOException | URISyntaxException e) {
         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .build();
      }
   }
}
```

Referências

- Amazon Simple Storage Service. Disponível em: https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html
- SPRING IO. Uploading Files. Disponível em: https://spring.io/guides/gs/uploading-files/