

Compile-time detection of machine image sniping

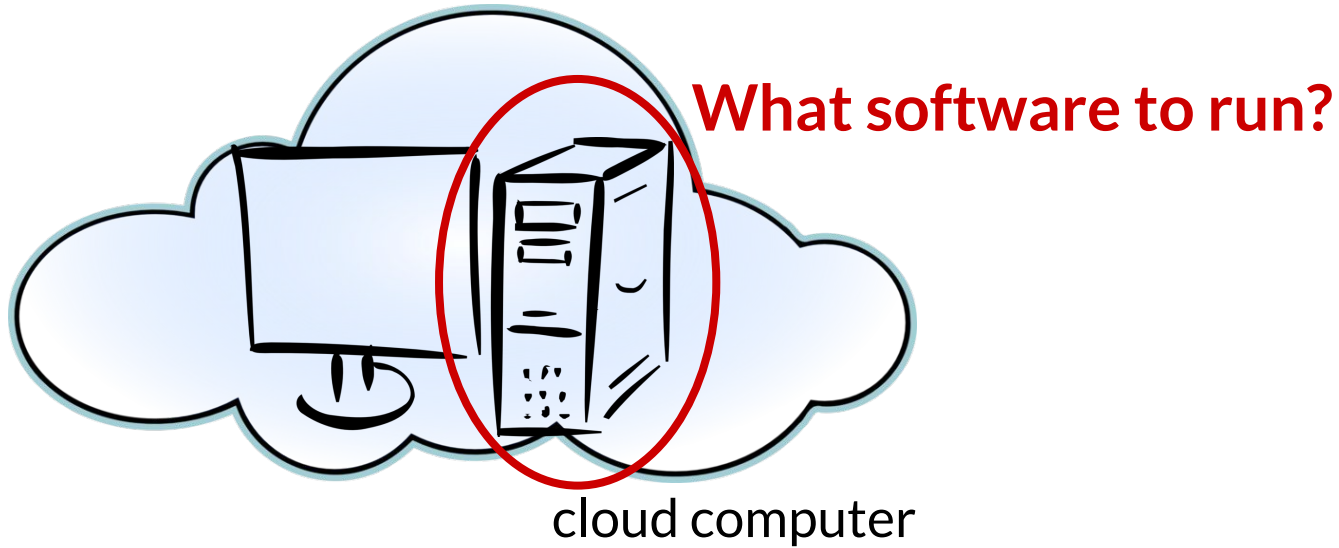
Martin Kellogg
University of Washington

What is a machine image?

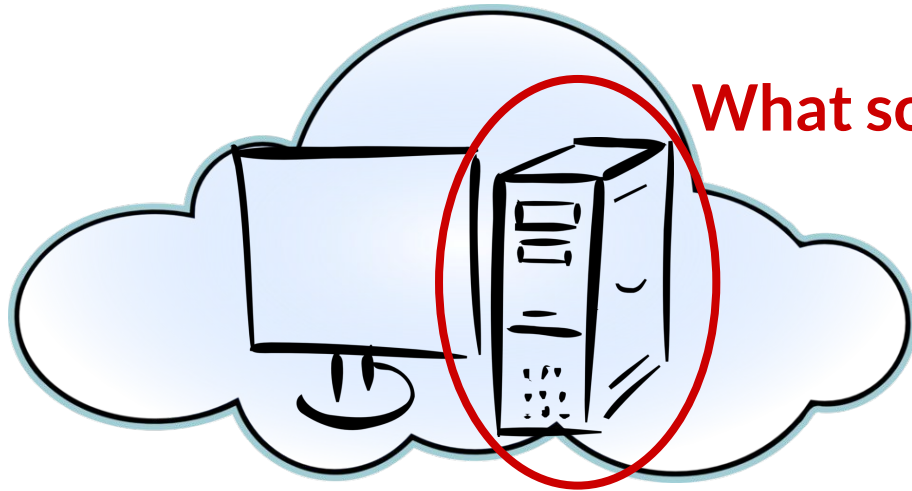


cloud computer

What is a machine image?



What is a machine image?



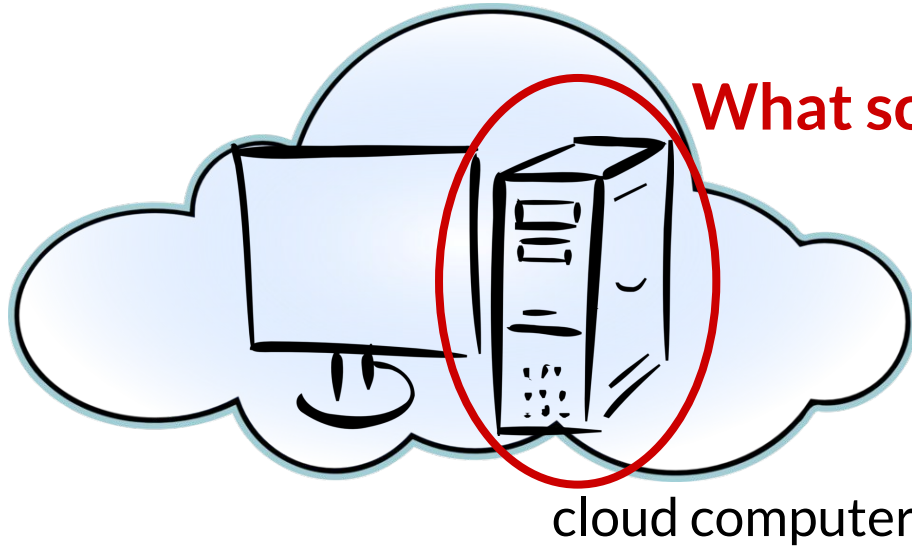
What software to run?

cloud computer



“machine image”

What is a machine image?



What software to run?

This software!



How to choose a machine image:

Look it up in a repository.

- By unique id:

```
aws ec2 describe-images --imageIds ami-5731123e
```

- By owner and name:

```
aws ec2 describe-images --owners myOrg \  
  --filters "Name=name,Values=ubuntu16.04-*"
```

- By name alone:

```
aws ec2 describe-images \  
  --filters "Name=name,Values=ubuntu16.04-*"
```

How to choose a machine image:

Look it up in a repository.

- By unique id:

```
aws ec2 describe-images --imageIds ami-5731123e
```

- By owner and name:

```
aws ec2 describe-images --owners myOrg \  
  --filters "Name=name,Values=ubuntu16.04-*"
```

- By name alone:

```
aws ec2 describe-images \  
  --filters "Name=name,Values=ubuntu16.04-*"
```



How to choose a machine image:

Look it up in a repository.

- By unique id:

```
aws ec2 describe-images --imageIds ami-5731123e
```



- By owner and name:

```
aws ec2 describe-images --owners myOrg \  
  --filters "Name=name,Values=ubuntu16.04-*"
```



- By name alone:

```
aws ec2 describe-images \  
  --filters "Name=name,Values=ubuntu16.04-*"
```


How to choose a machine image:

Look it up in a repository.

- By unique id:

```
aws ec2 describe-images --imageIds ami-5731123e
```



- By owner and name:

```
aws ec2 describe-images --owners myOrg \  
  --filters "Name=name,Values=ubuntu16.04-*"
```



- By name alone:

```
aws ec2 describe-images \  
  --filters "Name=name,Values=ubuntu16.04-*"
```



This isn't hypothetical...

Finding an AMI Using the AWS CLI

You can use AWS CLI commands for Amazon EC2 to list only the Linux AMIs that meet your needs. After locating an AMI that meets your needs, make note of its ID so that you can use it to launch instances. For more information, see [Launching an Instance Using the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

The `describe-images` command supports filtering parameters. For example, use the `--owners` parameter to display public AMIs owned by Amazon.

```
aws ec2 describe-images --owners self amazon
```



You can add the following filter to the previous command to display only AMIs backed by Amazon EBS:

```
--filters "Name=root-device-type,Values=ebs"
```



Important

Omitting the `--owners` flag from the `describe-images` command will return all images for which you have launch permissions, regardless of ownership.

This isn't hypothetical...

Finding an AMI Using the AWS CLI

You can use AWS CLI commands for Amazon EC2 to list only the Linux AMIs that meet your needs. After locating an AMI that meets your needs, make note of its ID so that you can use it to launch instances. For more information, see [Launching an Instance Using the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

The `describe-images` command supports filtering parameters. For example, use the `--owners` parameter to display public AMIs owned by Amazon.

```
aws ec2 describe-images --owners self amazon
```

You can add the following filter to the previous command to display only AMIs backed by Amazon EBS:

```
--filters "Name=root-device-type,Values=ebs"
```

Important

Omitting the `--owners` flag from the `describe-images` command will return all images for which you have launch permissions, regardless of ownership.

This isn't hypothetical...

```
DescribeImagesRequest request = new DescribeImagesRequest();  
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));  
api.describeImages(request);
```

This isn't hypothetical...

```
DescribeImagesRequest request = new DescribeImagesRequest();  
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));  
api.describeImages(request);
```



This isn't hypothetical...

```
DescribeImagesRequest request = new DescribeImagesRequest();  
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));  
api.describeImages(request);
```



**Unsafe: returns all
images with that name
from public repo!**

How to make this client safe?

```
DescribeImagesRequest request = new DescribeImagesRequest();  
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));  
api.describeImages(request);
```

How to make this client safe?

```
DescribeImagesRequest request = new DescribeImagesRequest();  
request.withFilters(new Filter("name", "RHEL-7.5_HVM_GA"));  
request.withOwners("myOrg");  
api.describeImages(request);
```

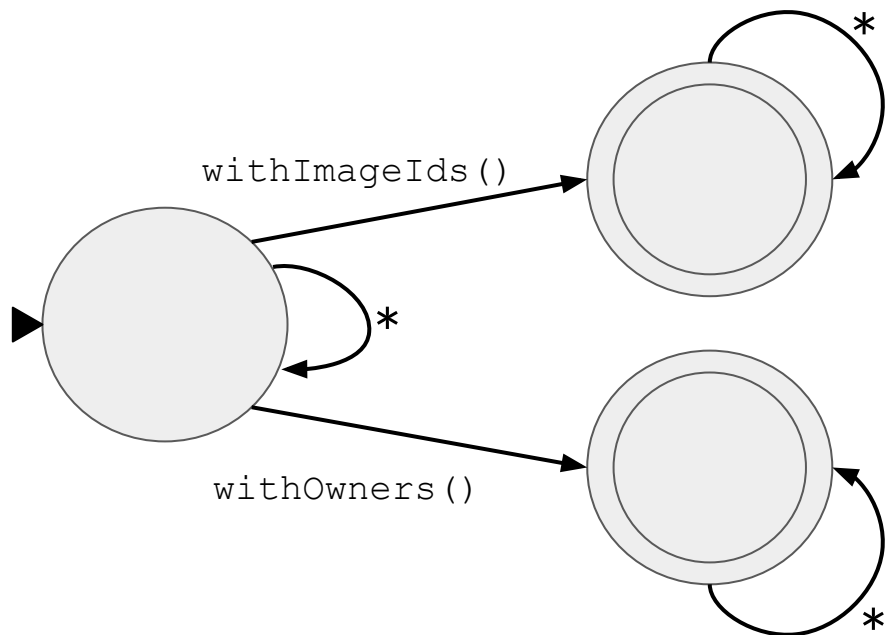

How to prove this safe?

How to prove this safe?

A traditional approach: **typestate**

How to prove this safe?

A traditional approach: **typestate**



How to prove this safe?

A traditional approach: **typestate**

- create a finite state machine for each object
- on method calls, transition the state machine
- only permit certain calls in certain states
- use alias analysis to ensure all copies are in same state

How to prove this safe?

A traditional approach: **typestate**

- create a finite state machine for each object
- on method calls, transition the state machine
- only permit certain calls in certain states
- ~~● use alias analysis to ensure all copies are in same state~~

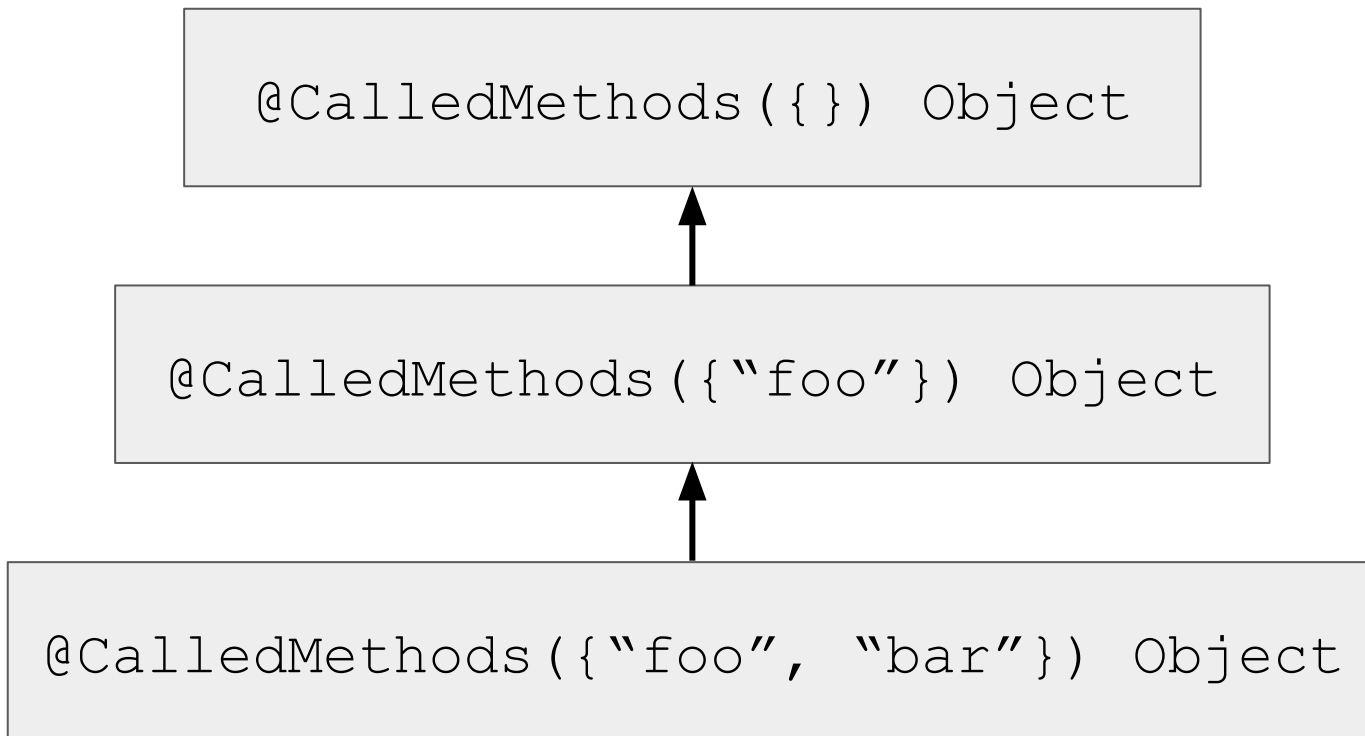
Advantages of a type system

- still provides a proof
- modular \Rightarrow scalable
- no alias analysis \Rightarrow cheap

Specifying describeImages ()

```
DescribeImageResponse describeImages (  
    @CalledMethods ("withImageIds || withOwners")  
    DescribeImageRequest request) {  
  
    ...  
  
}
```

Type hierarchy



Experimental results

No. projects	548
Source LoC	9.2M
True positives	14
False positives	3

Example: Netflix/SimianArmy

```
public List<Image> describeImages (String... imageIds) {  
    DescribeImagesRequest request =  
        new DescribeImagesRequest ();  
  
    if (imageIds != null) {  
        request.setImageIds (Arrays.asList (imageIds));  
    }  
  
    DescribeImagesResult result =  
        ec2client.describeImages (request);  
  
    return result.getImages ();  
}
```

Accumulation analysis

- Our type system *accumulates* method calls

Accumulation analysis

- Our type system *accumulates* method calls

Insight: can generalize to any analysis that accumulates something

Accumulation analyses

- machine sniping (this talk!)

Accumulation analyses

- machine sniping (this talk!)
- the builder pattern

Accumulation analyses

- machine sniping (this talk!)
- the builder pattern
- dependency injection providers

Contributions

- Accumulation analysis can detect machine-image sniping vulnerabilities -- and more
- Experiments that show:
 - those vulnerabilities exist in practice, and
 - we can find them!

Lombok/AutoValue builders

Lombok and AutoValue generate builder implementations from structs

Fields can be marked `@NonNull`; NPE if the corresponding setter isn't called

Lombok/AutoValue builders

@Builder

```
public class UserIdentity {  
    private final @NonNull String name;  
    private final @NonNull String displayName;  
    private final @NonNull ByteArray id;  
}
```

Lombok/AutoValue builders

```
UserIdentity identity =  
    UserIdentity.builder()  
        .name(username)  
        .displayName(displayName)  
        .id(generateRandom(32))  
        .build();
```

Lombok/AutoValue builders

```
UserIdentity identity =  
    UserIdentity.builder()  
        .name(username)  
        .displayName(displayName) // (optional)  
        .id(generateRandomId())  
        .build();
```




Lombok/AutoValue builders

```
UserIdentity identity =  
    UserIdentity.builder()  
        .name(username)  
        // .displayName(displayName)  
        .id(generateRandom(32))  
        .build();
```

Lombok/AutoValue builders

```
UserIdentity identity =  
    UserIdentity.builder()  
        .name (use  
        // .display (use (displayName)  
        .id (generom (32))  
        .build ();
```



Lombok user study

6 industrial developers with Java + Lombok experience

Task: add a new `@NonNull` field to a builder, and update all call sites

Results:

- 6/6 succeeded with our tool, only 3/6 without
- Those who succeeded at both 1.5x faster with our tool
- *“It was easier to have the tool report issues at compile time”*

Lombok/AutoValue case studies

5 projects: 2 Lombok, 3 AutoValue (~500k sloc)

563 calls verified, 1 true positive (google/gapic-generator)

110 annotations, 19 false positives