A Placement Vulnerability Study in Multi-tenant Public Clouds

Venkat(anathan) Varadarajan, Yinqian Zhang, Thomas Ristenpart and Michael Swift
venkatv@cs.wisc.edu

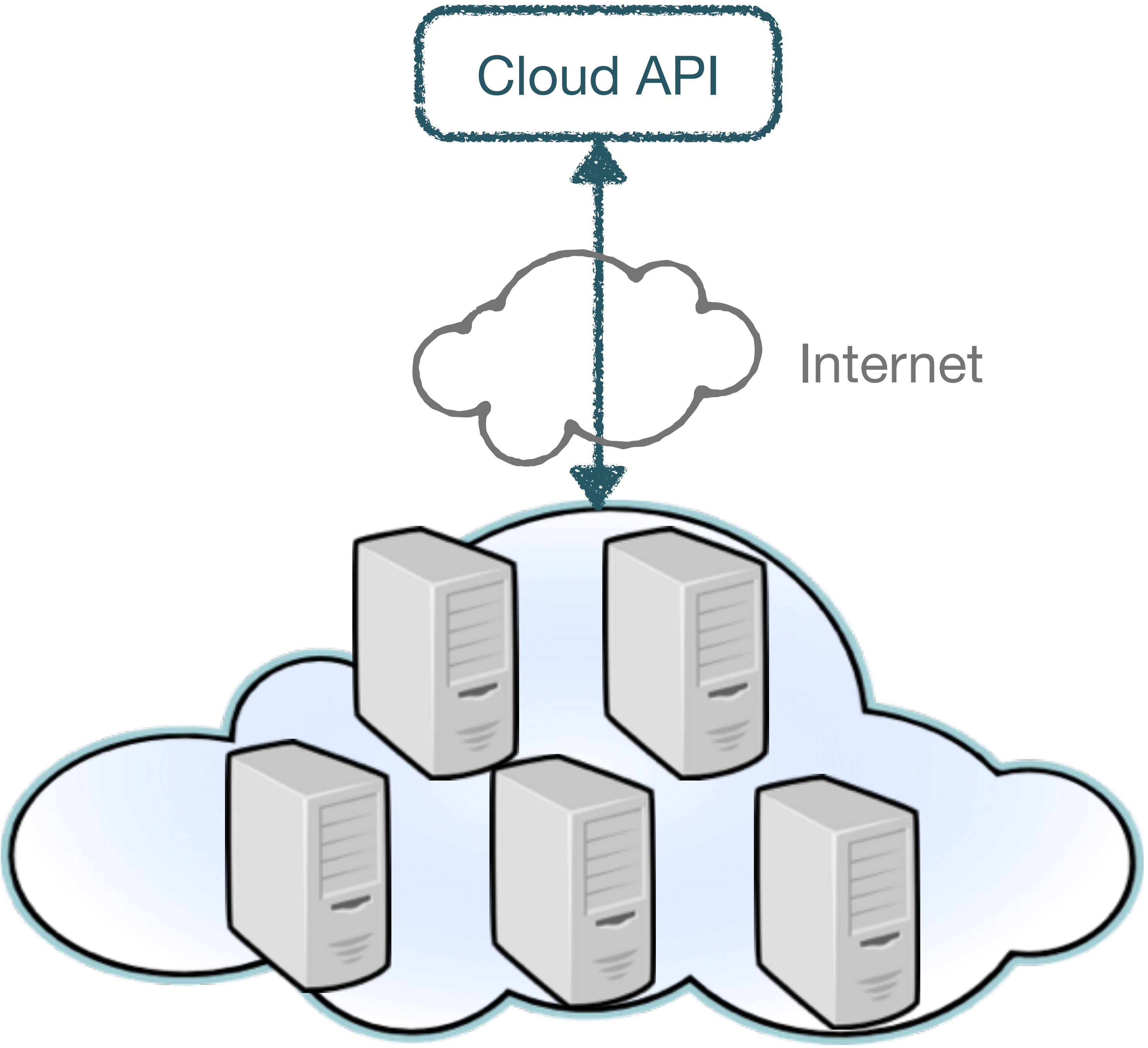
DEPARTMENT OF Computer Sciences UNIVERSITY OF WISCONSIN-MADISON

THE OHIO STATE UNIVERSITY

CORNELL TECH

USENIX Security 2015

Public Clouds

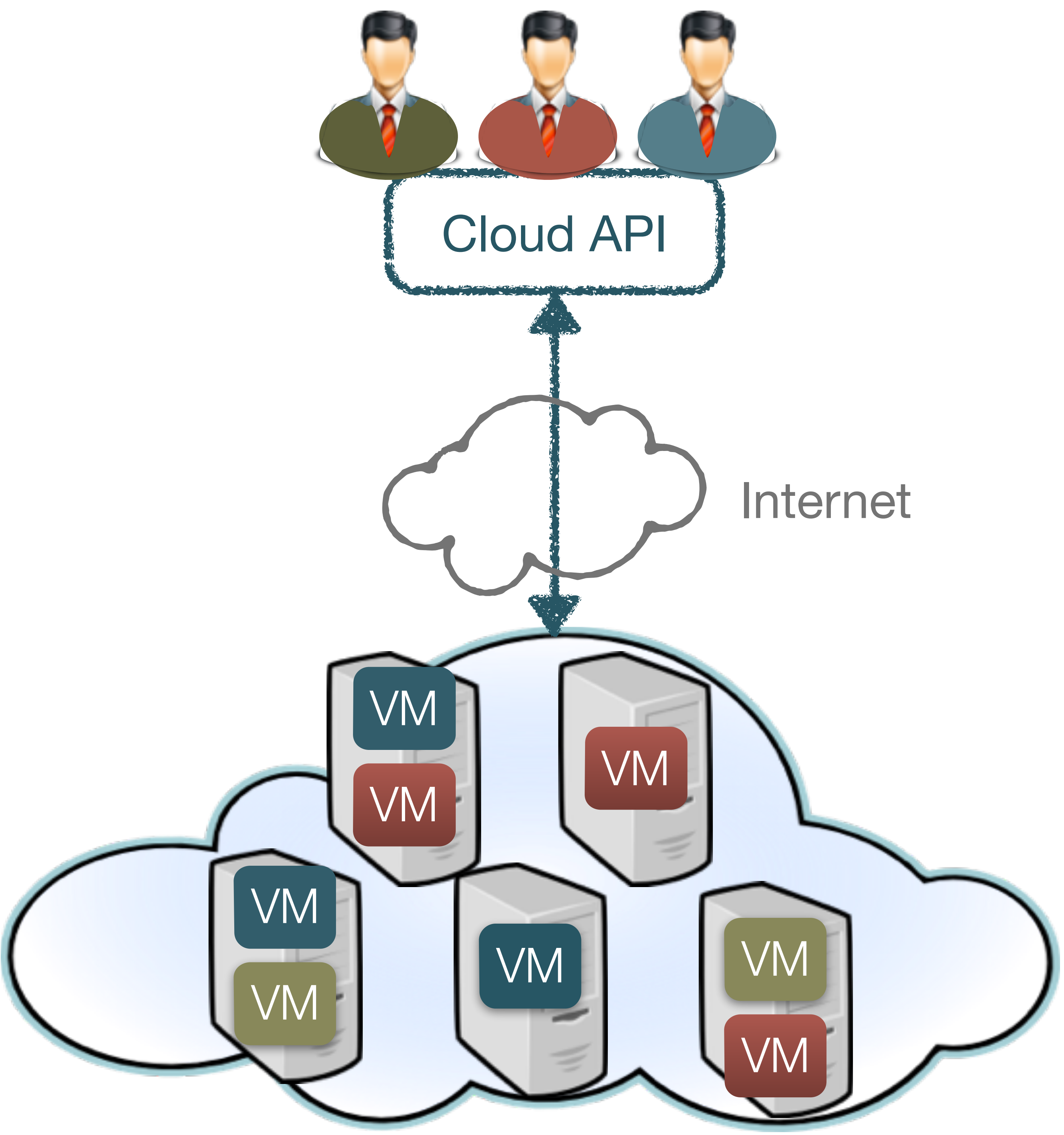


Google Compute Engine

amazon web services EC2

Microsoft Azure

Public Clouds

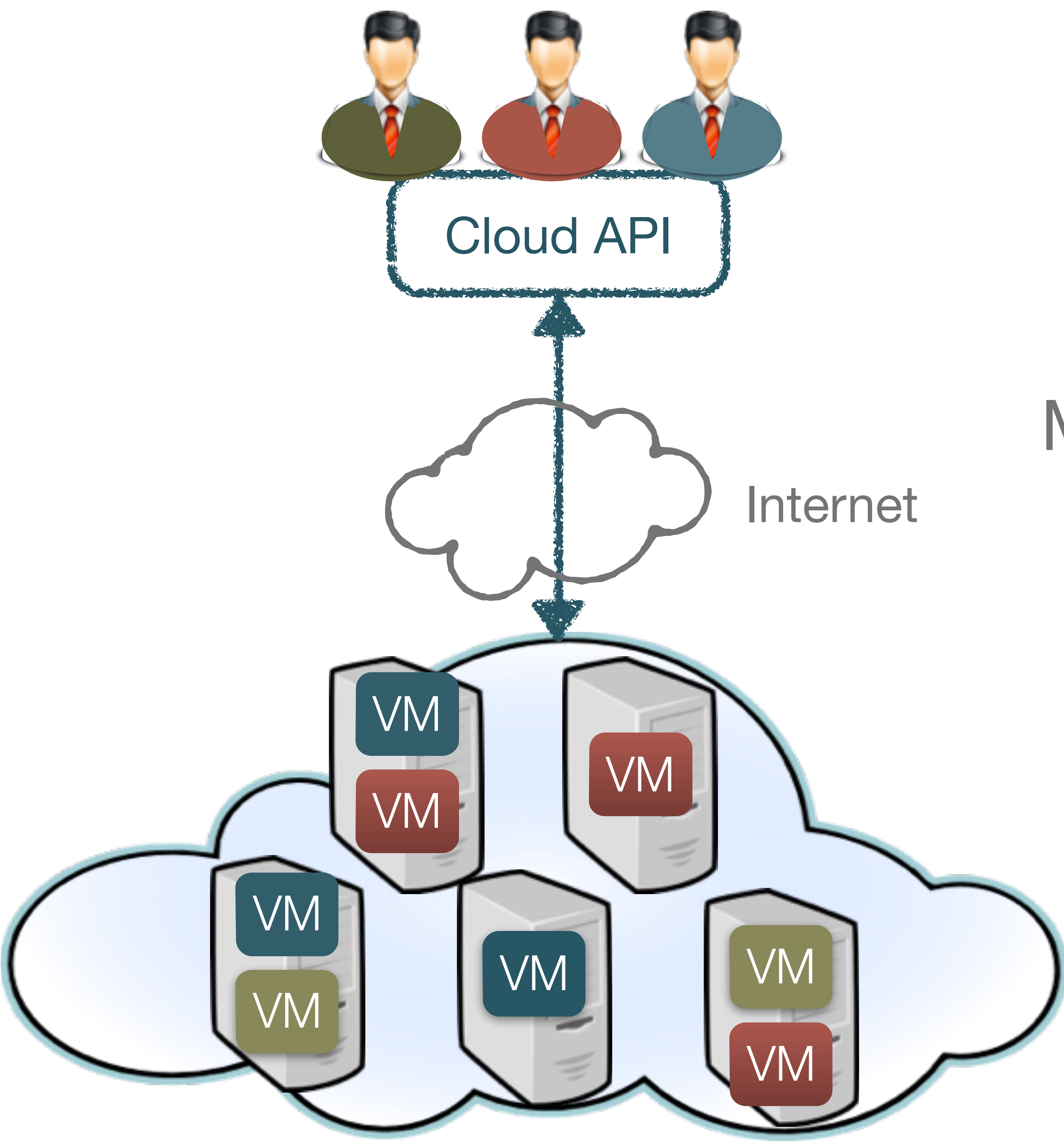


Google Compute Engine

amazon web services EC2

Microsoft Azure

Public Clouds



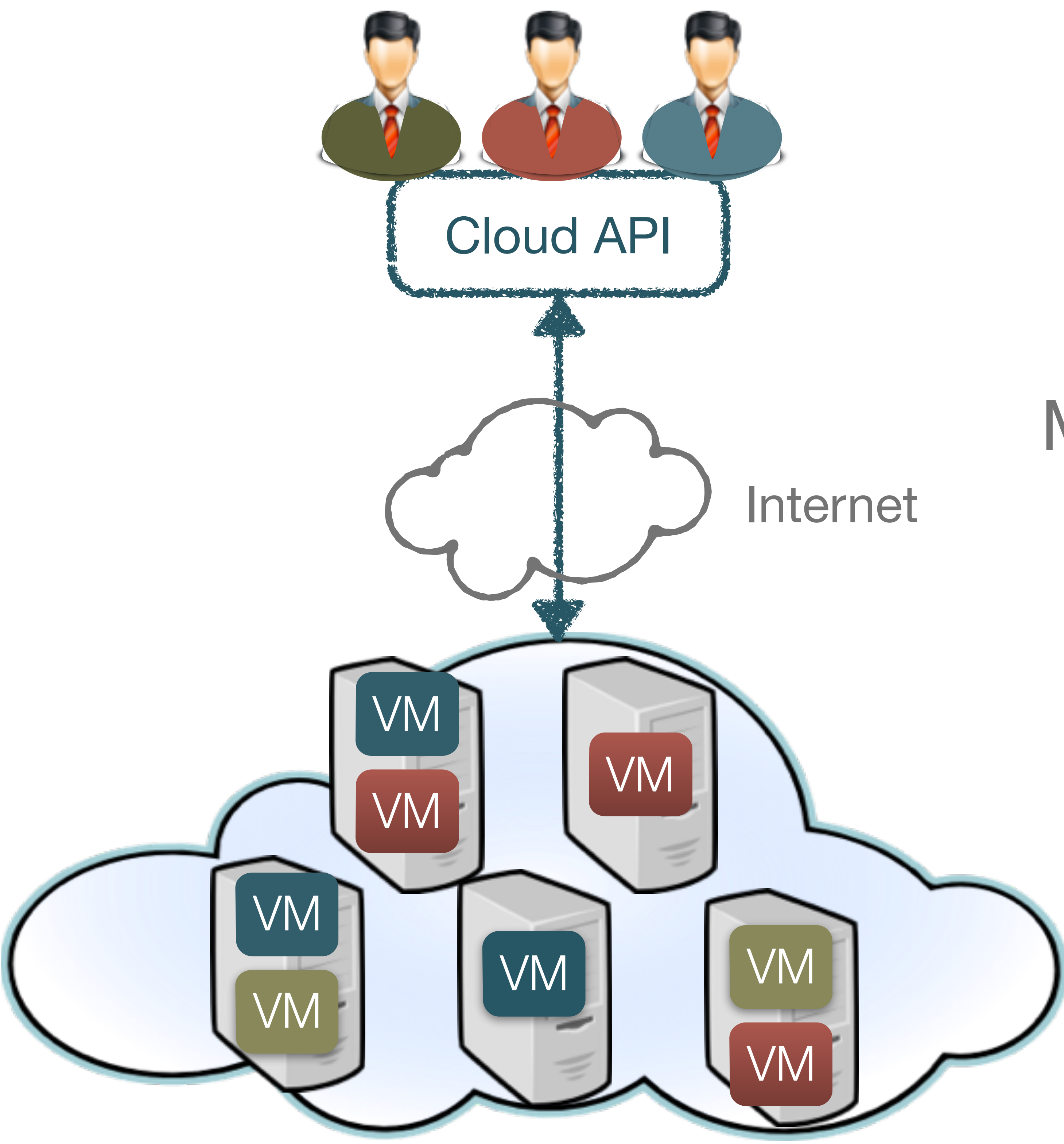
Google Compute Engine

amazon web services EC2

Microsoft Azure

Multi-tenancy

Public Clouds



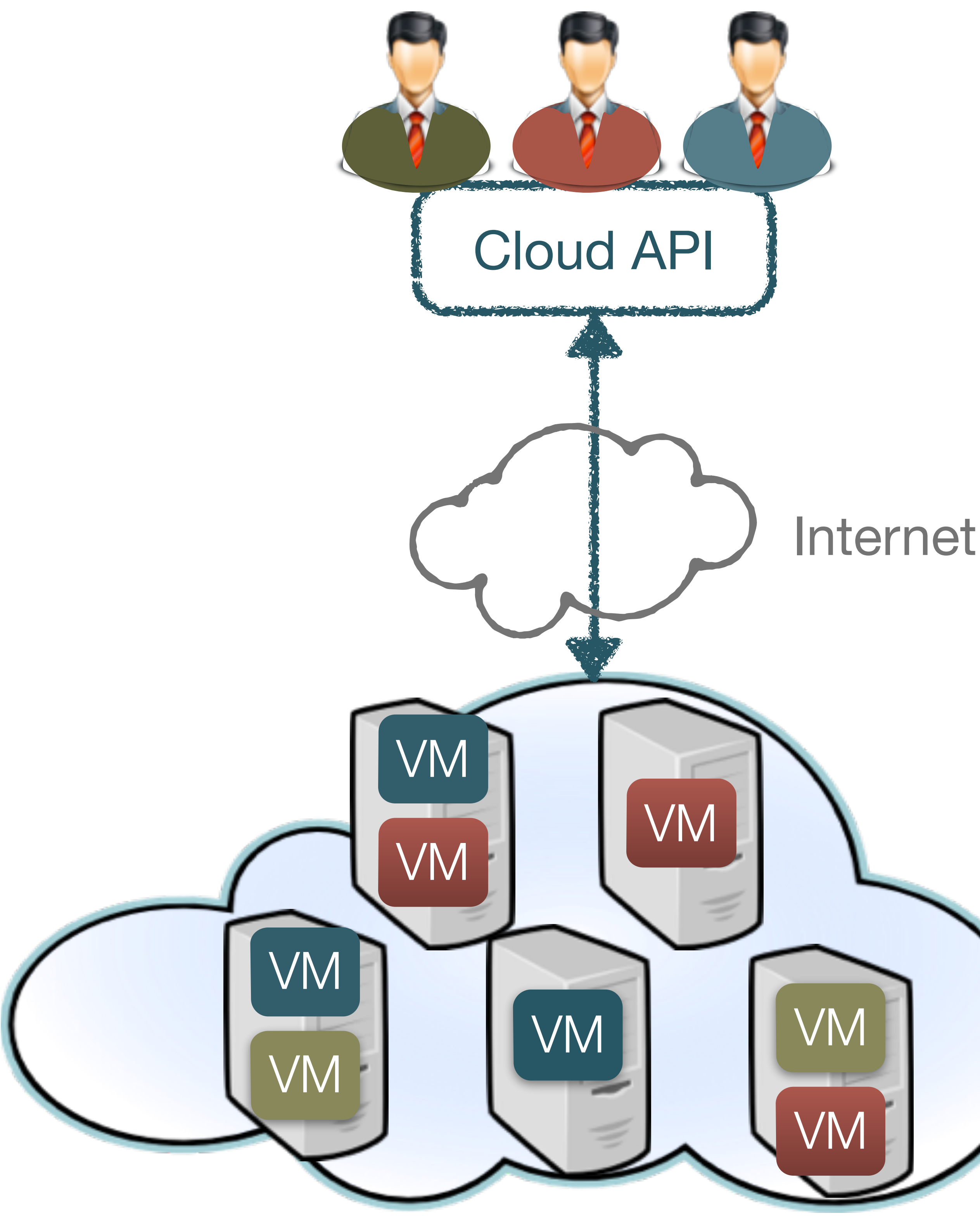
Google Compute Engine

amazon web services EC2

Microsoft Azure

Multi-tenancy + Public

Public Clouds



Google Compute Engine

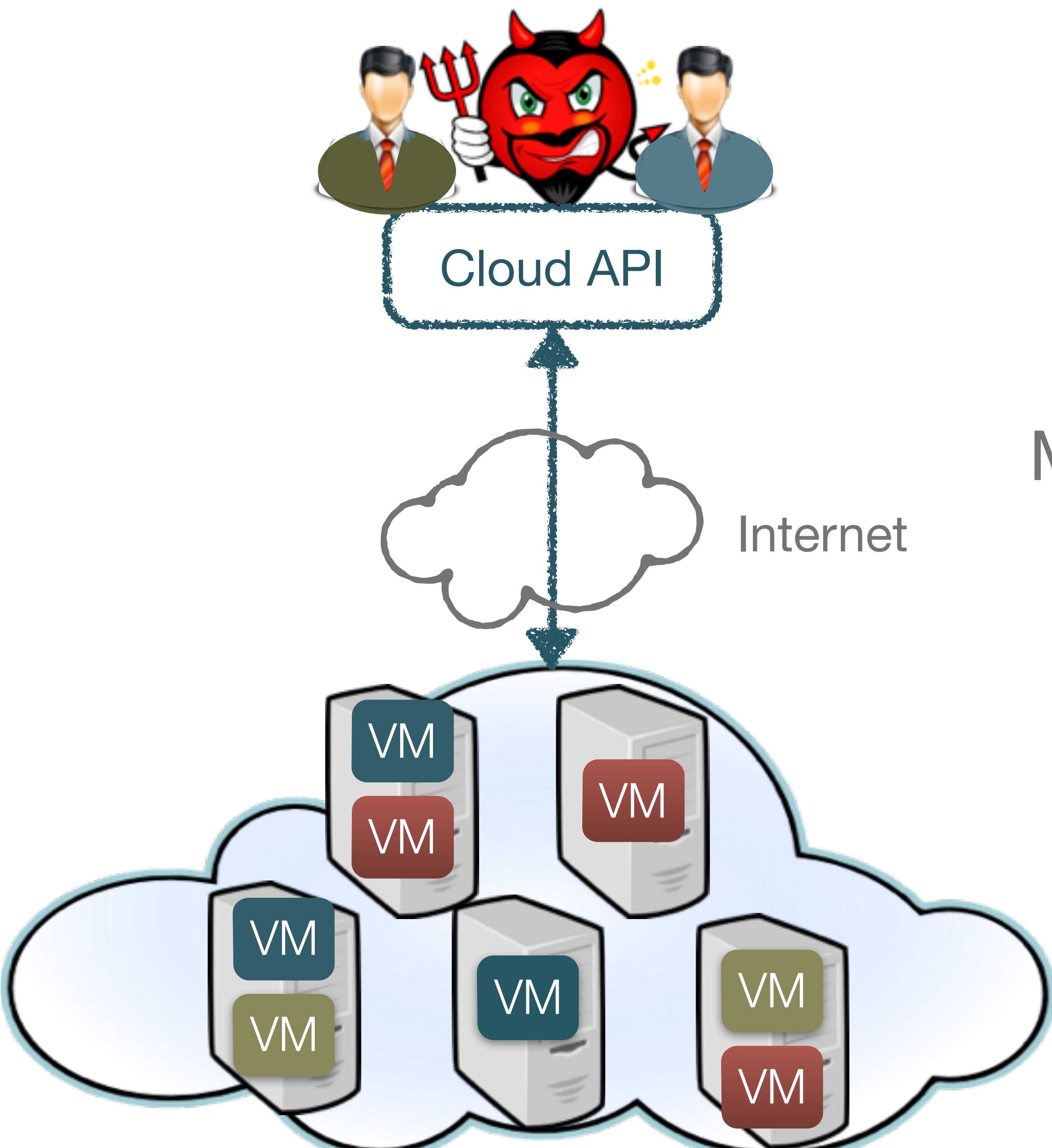
amazon web services™ EC2

Microsoft Azure

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location

Public Clouds



Google Compute Engine

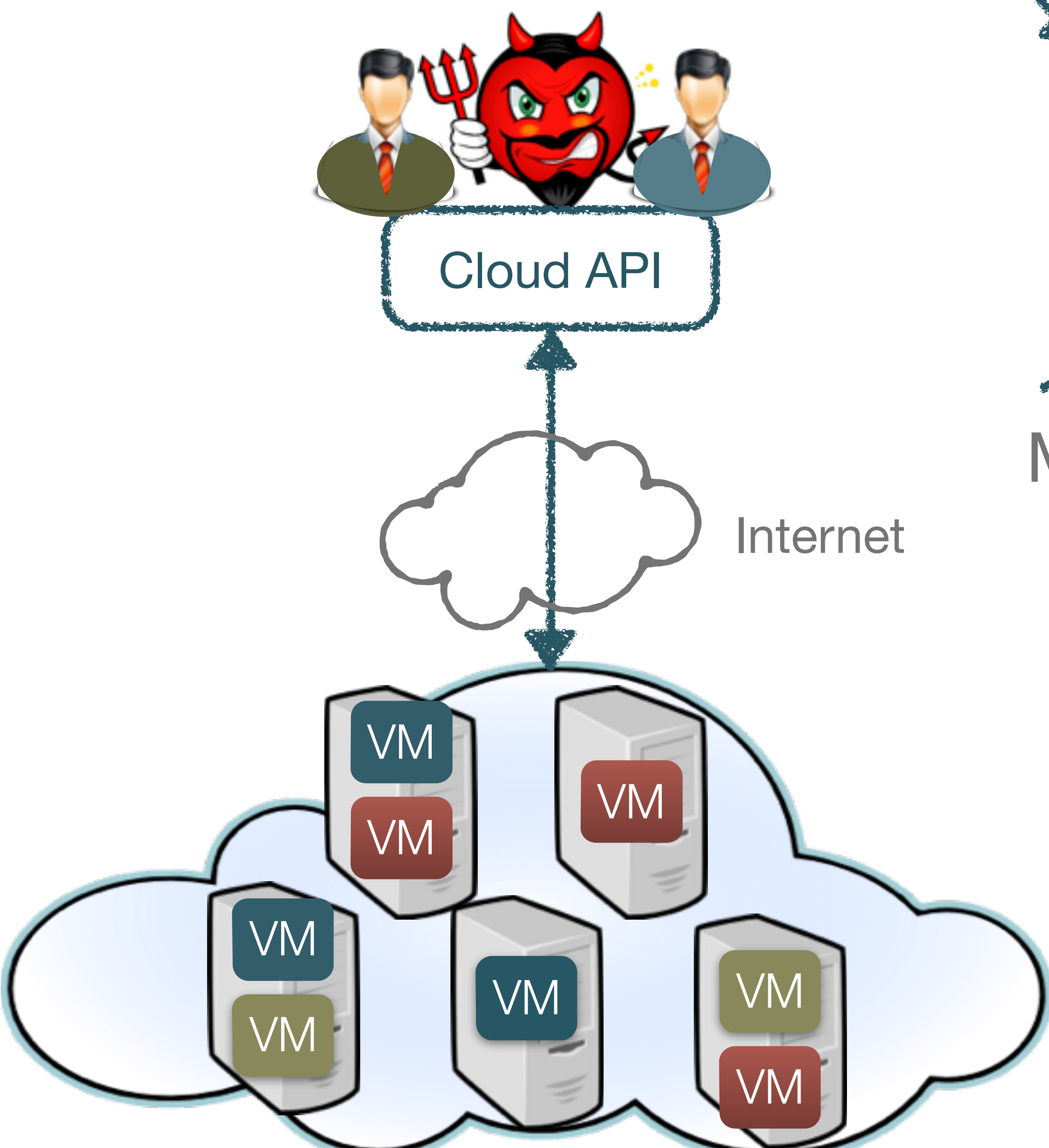
amazon web services™ EC2

Microsoft Azure

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location

Public Clouds



Google Compute Engine

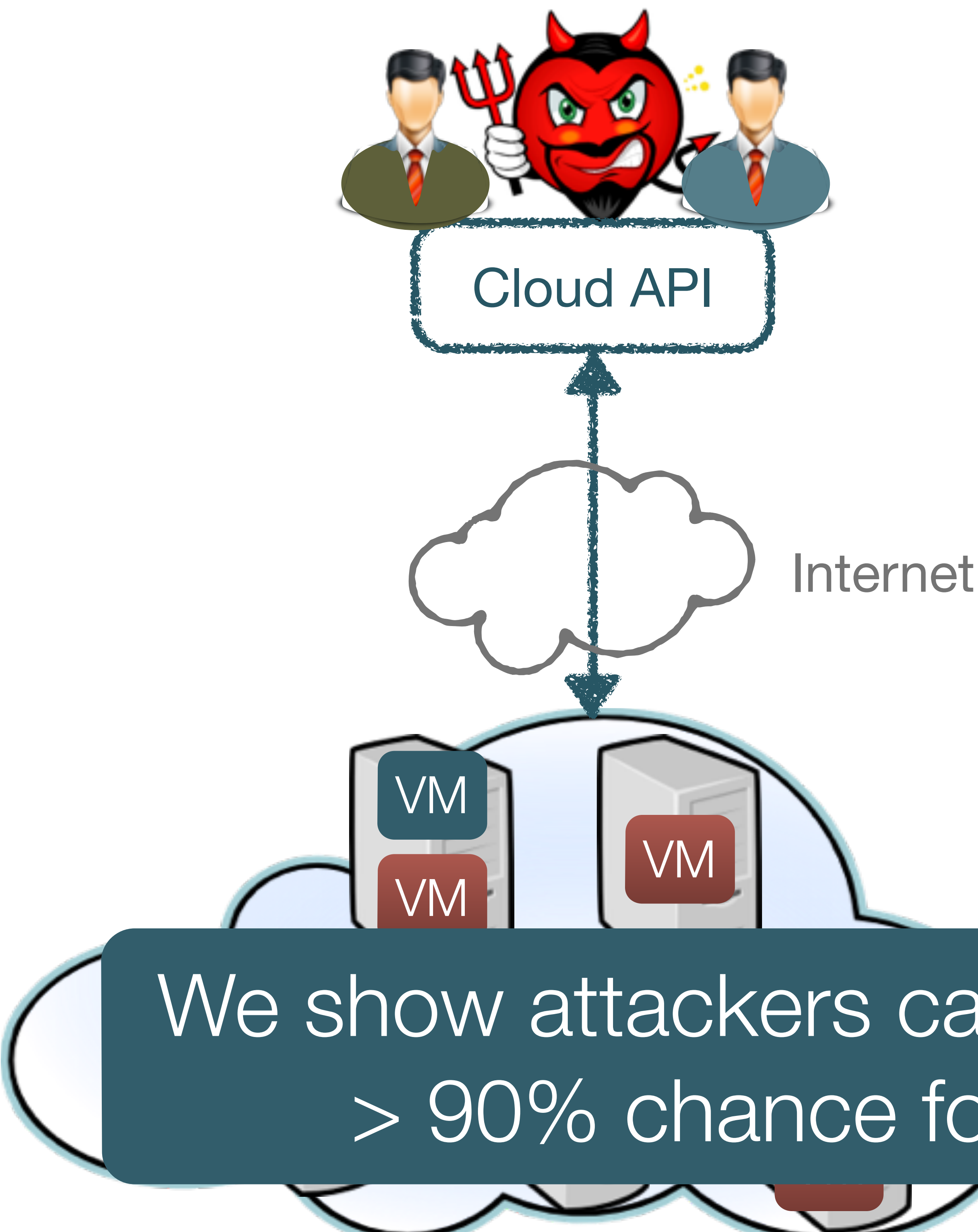
amazon web services EC2

Microsoft Azure

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location

Public Clouds



Google Compute Engine

amazon web services EC2

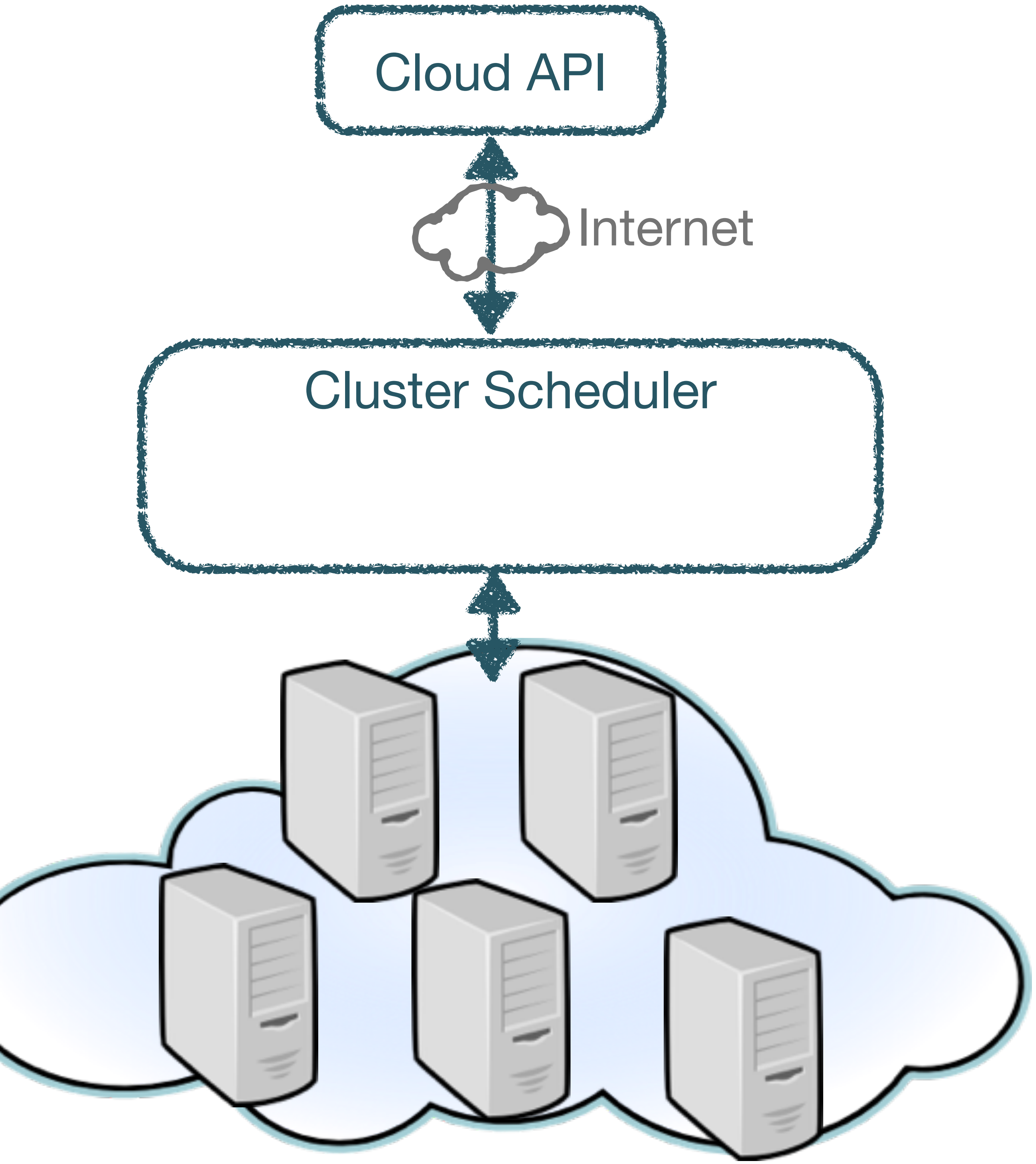
Microsoft Azure

Multi-tenancy + Public

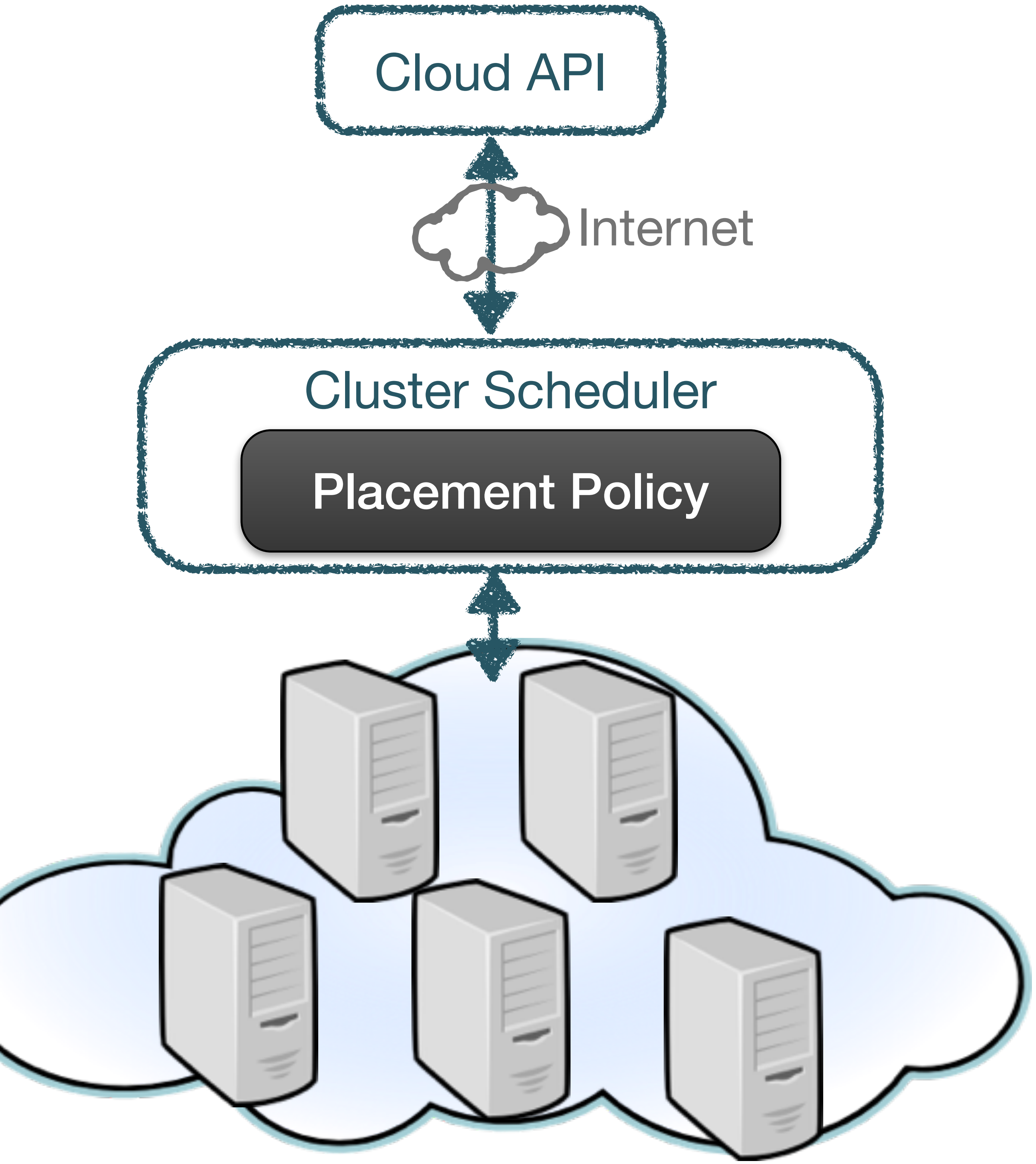
Big concern: cross-VM attacks via co-location

We show attackers can achieve co-location with > 90% chance for as low as *14* cents!

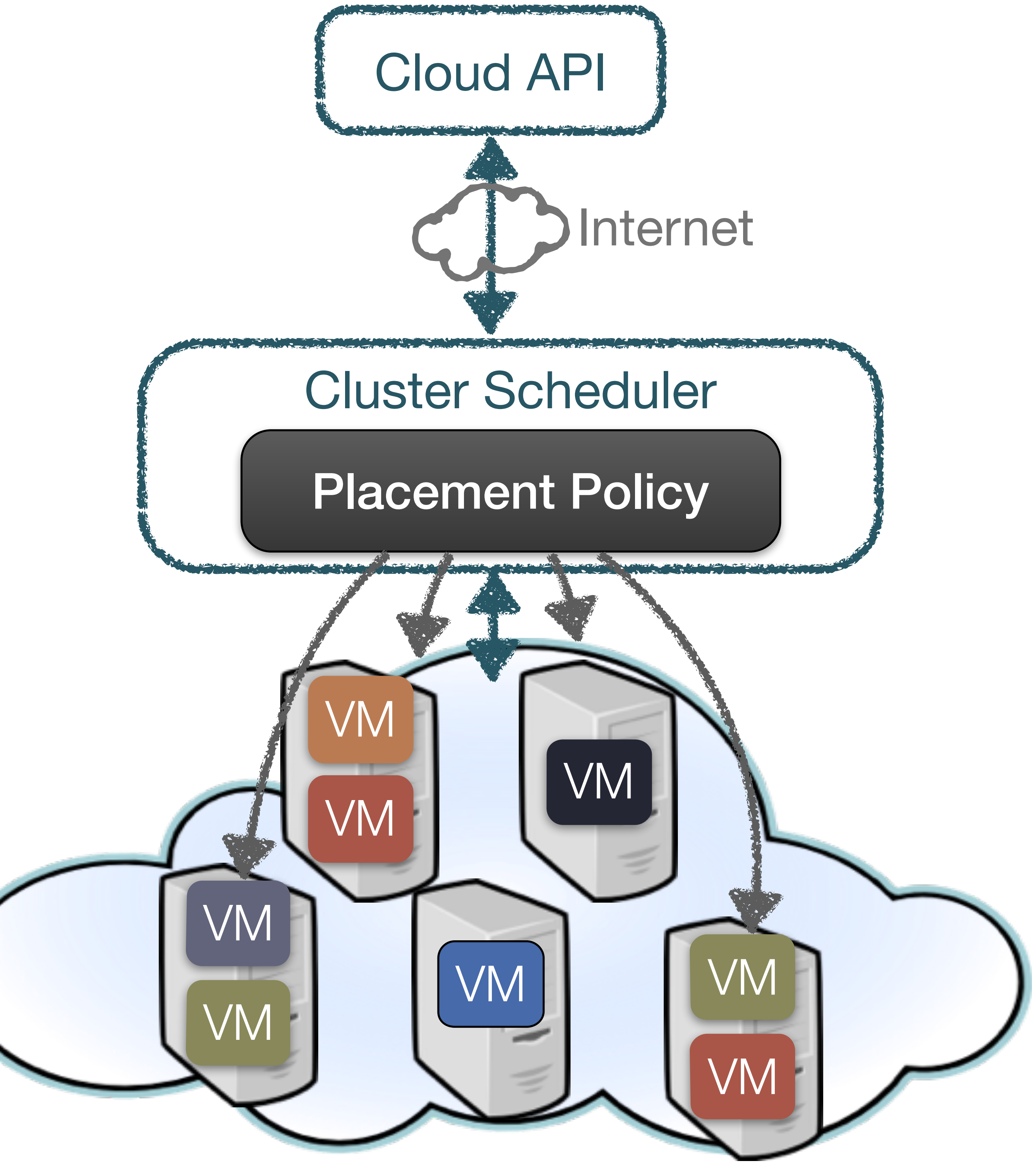
Placement and Isolation in Public Clouds



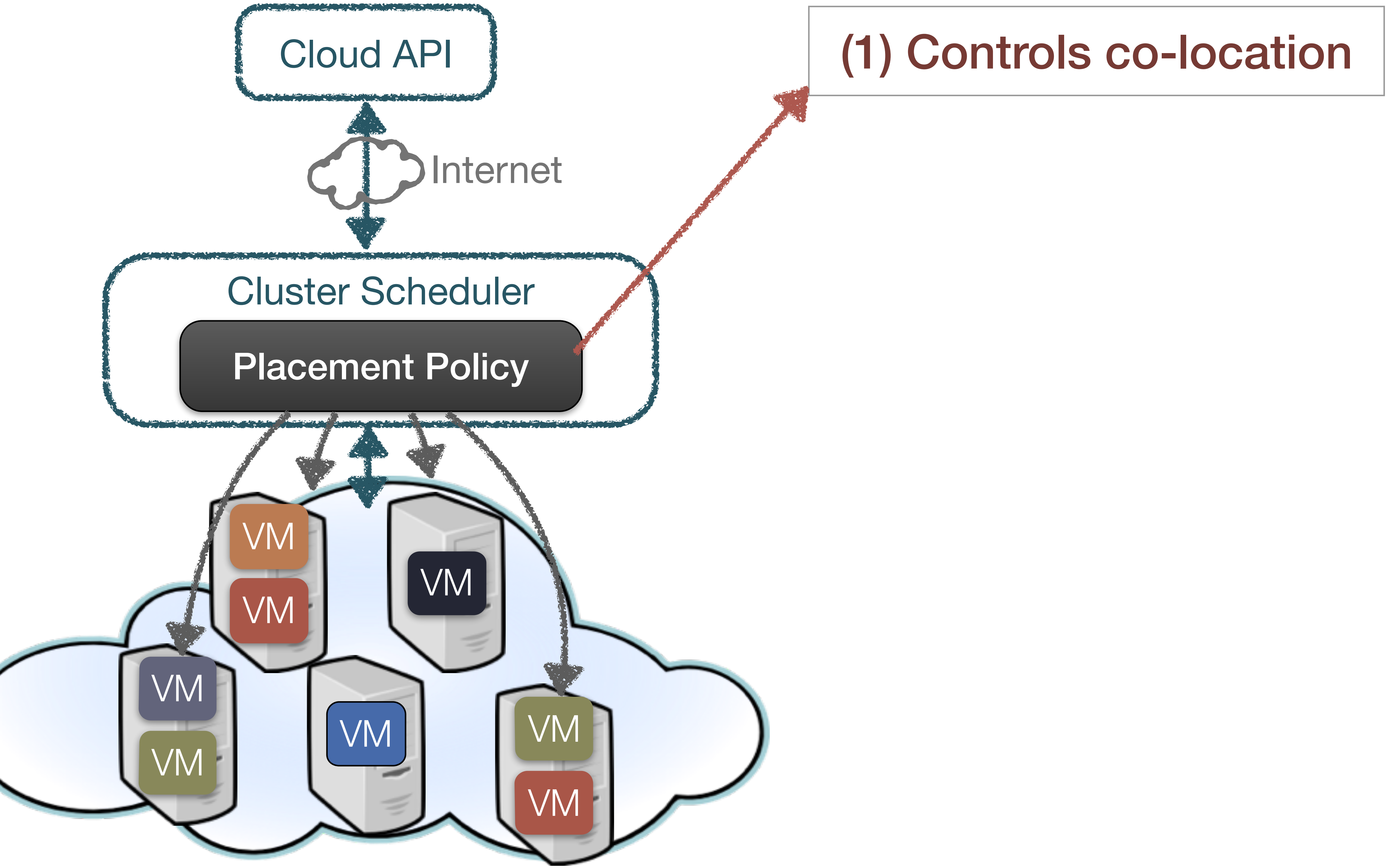
Placement and Isolation in Public Clouds



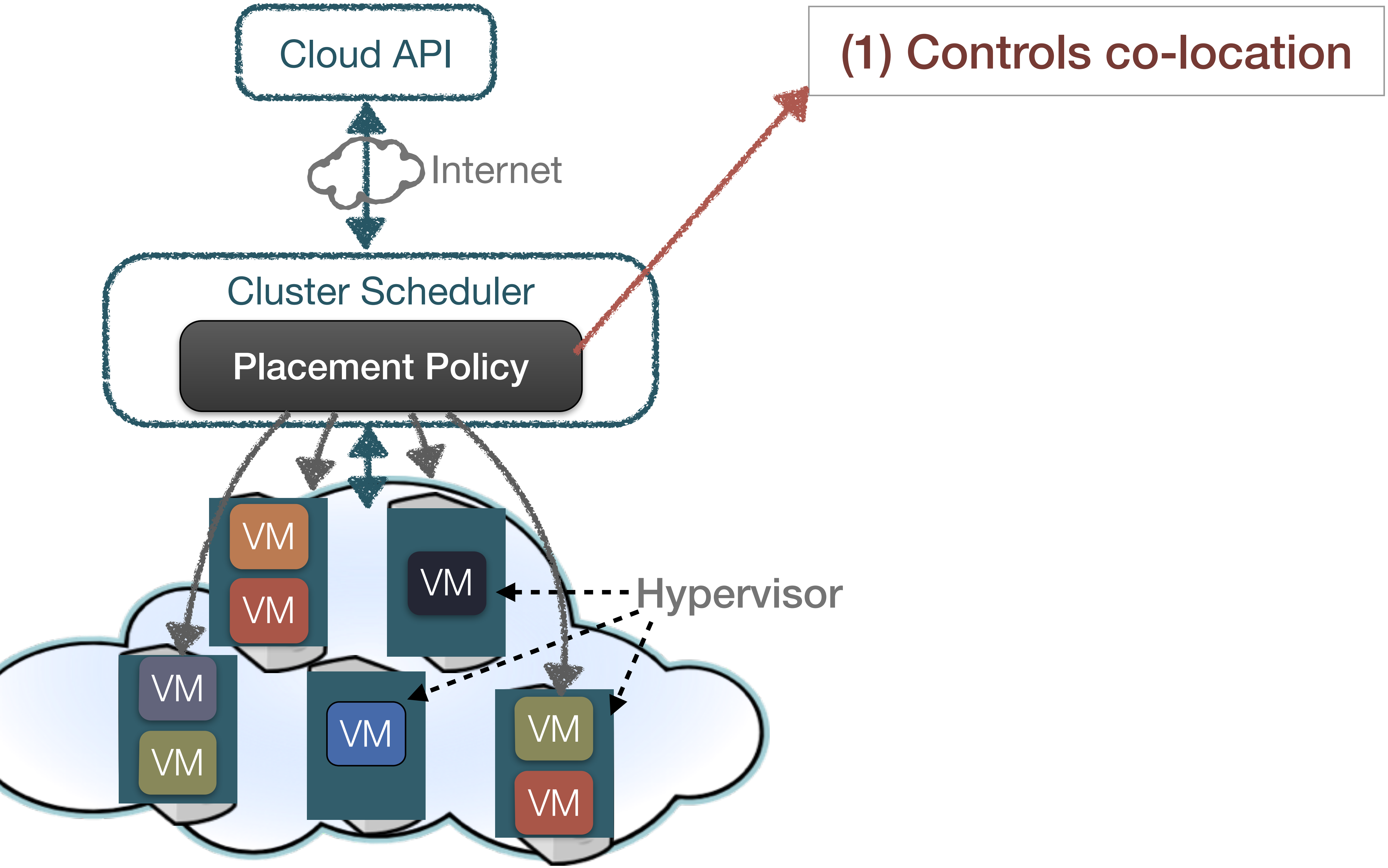
Placement and Isolation in Public Clouds



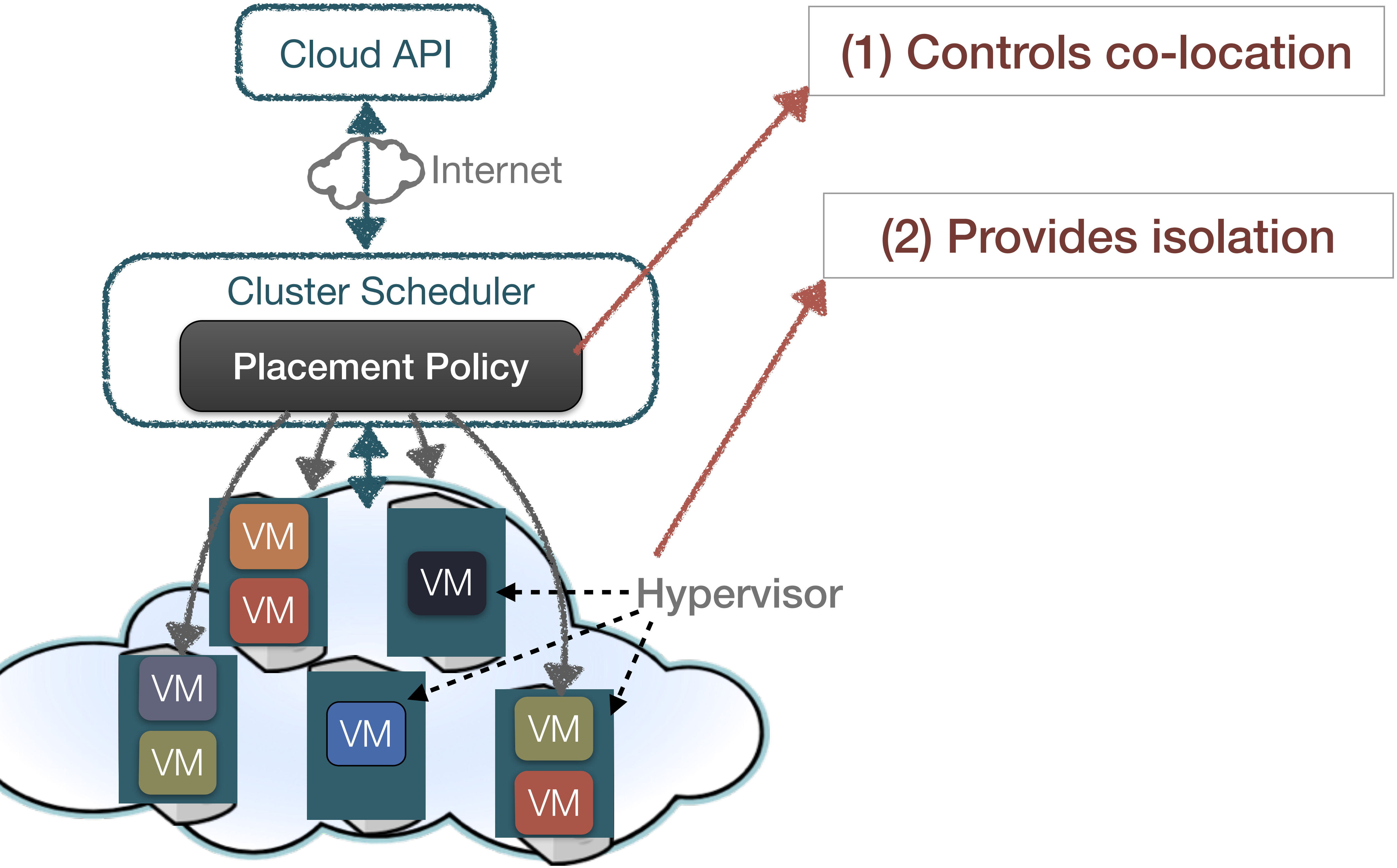
Placement and Isolation in Public Clouds



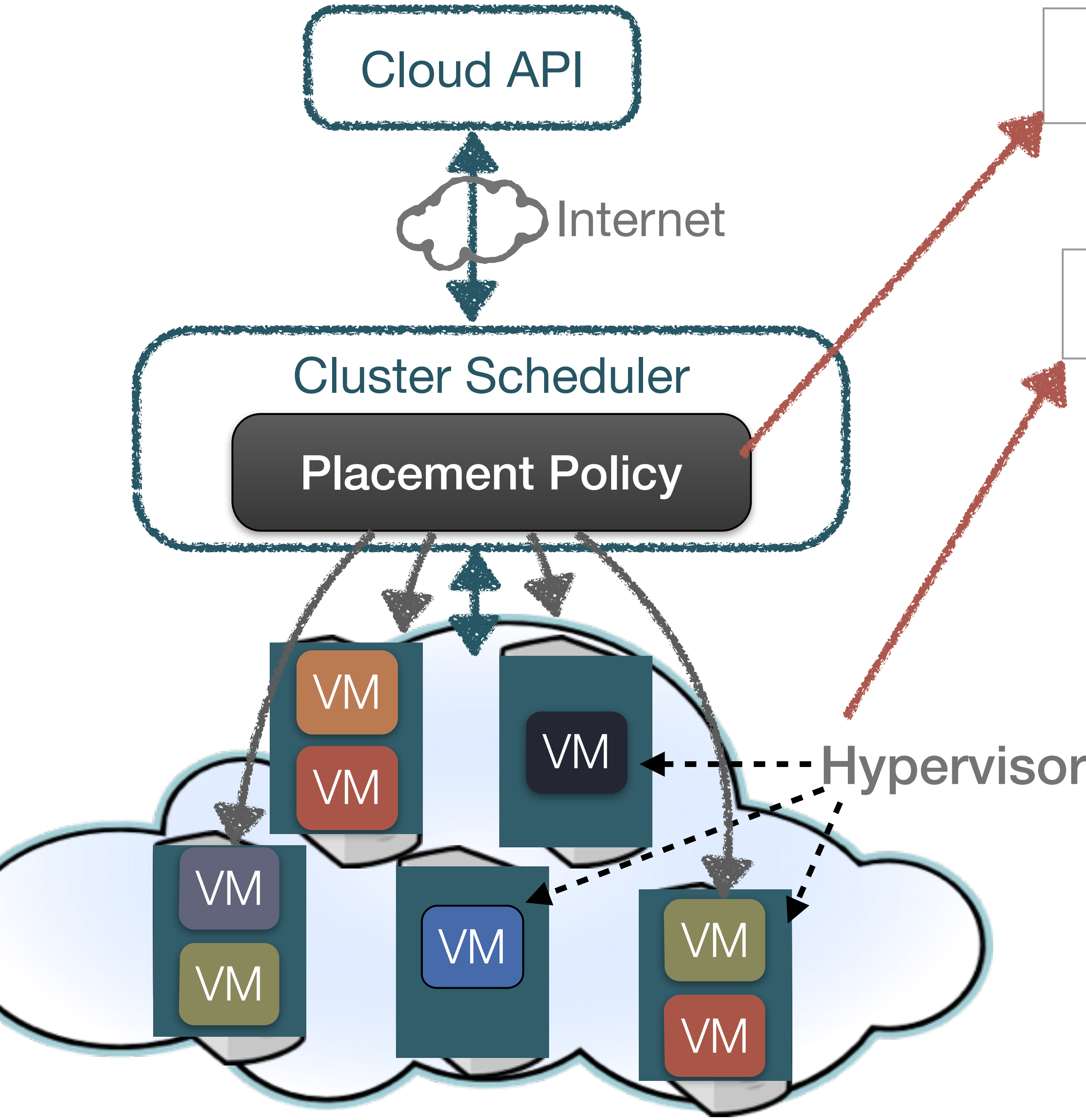
Placement and Isolation in Public Clouds



Placement and Isolation in Public Clouds



Placement and Isolation in Public Clouds

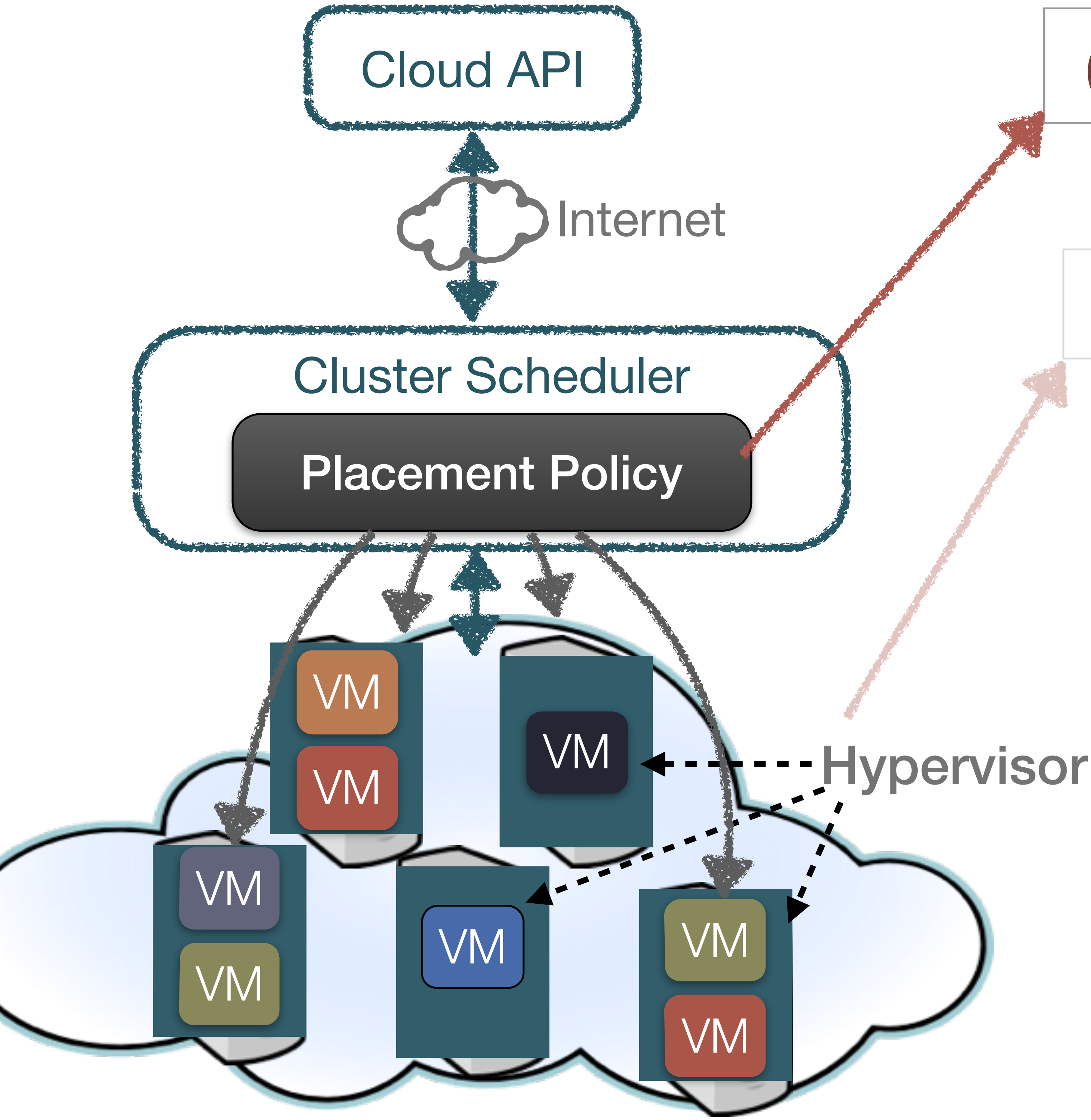


(1) Controls co-location

(2) Provides isolation

- **Cross-VM side- and covert-channels** [Ristenpart'09, Xu'11, Wu'12, Zhang'12, Yarom'14, Liu'15, Gruss'15]
- **Resource-stealing, DoS** [Zhou'11, Varadarajan'12]

Placement and Isolation in Public Clouds

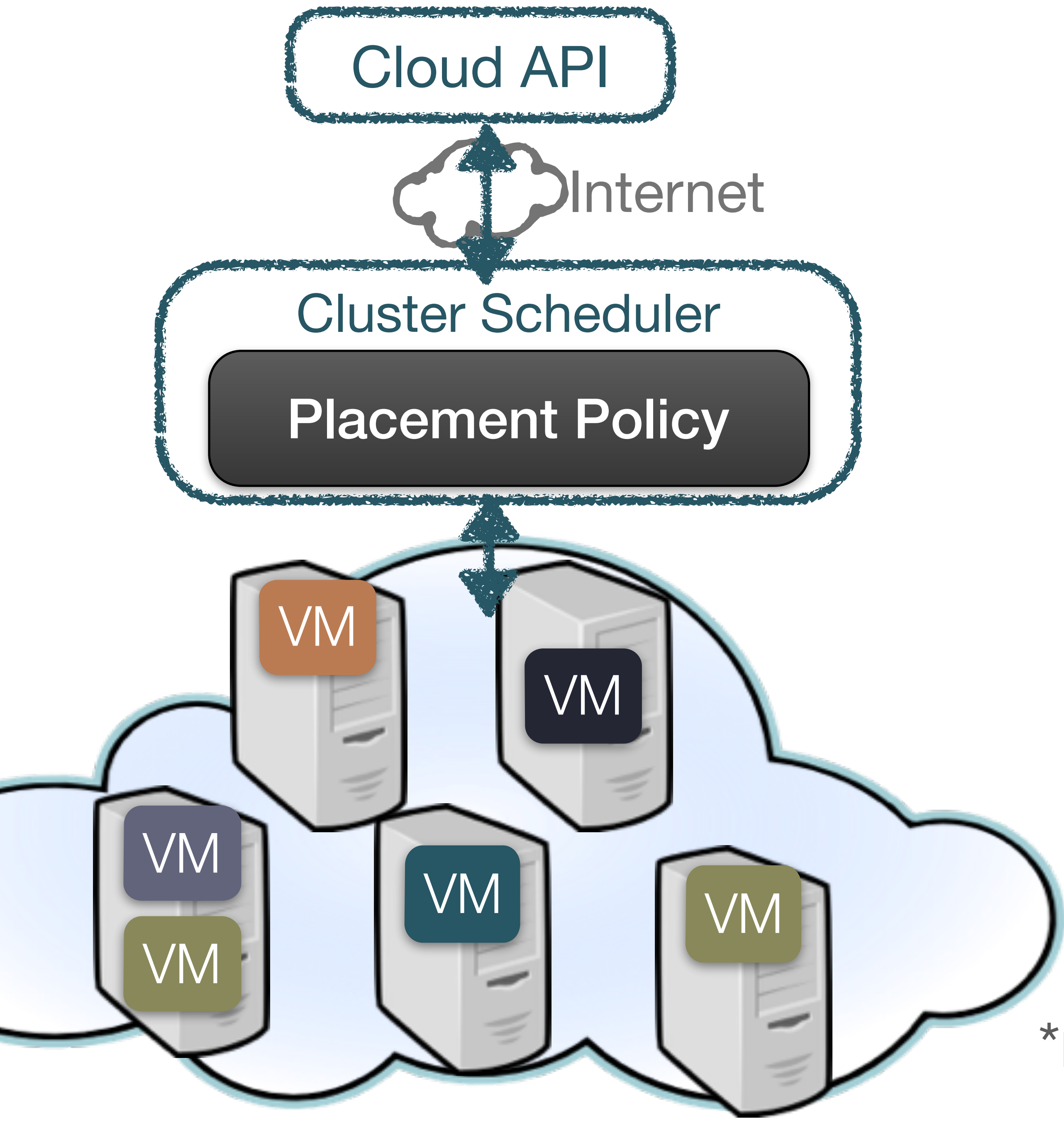


(1) Controls co-location

(2) Provides isolation

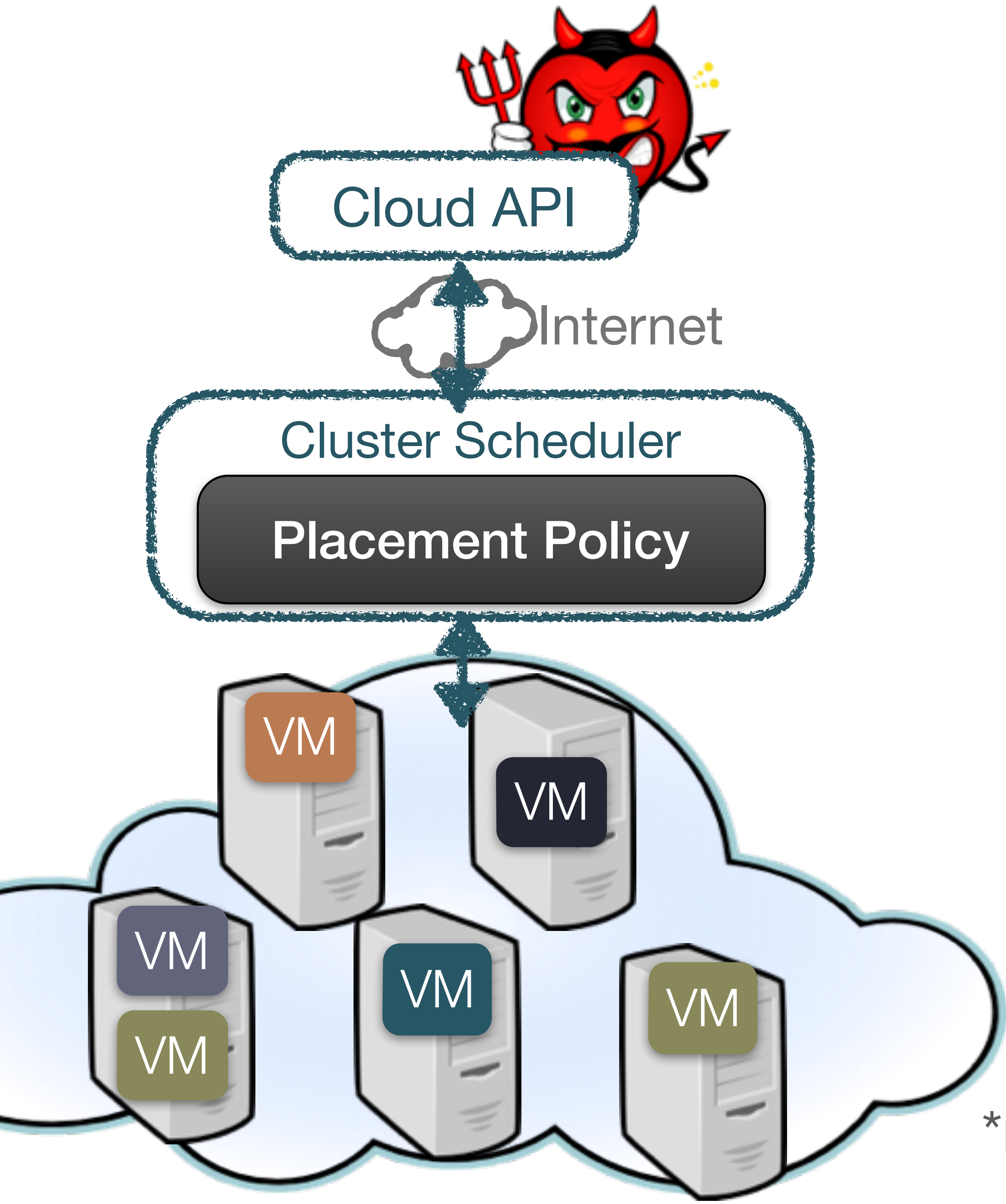
- **Cross-VM side- and covert-channels** [Ristenpart'09, Xu'11, Wu'12, Zhang'12, Yarom'14, Liu'15, Gruss'15]
- **Resource-stealing, DoS** [Zhou'11, Varadarajan'12]

Prior Work on Co-location: 2009 Study



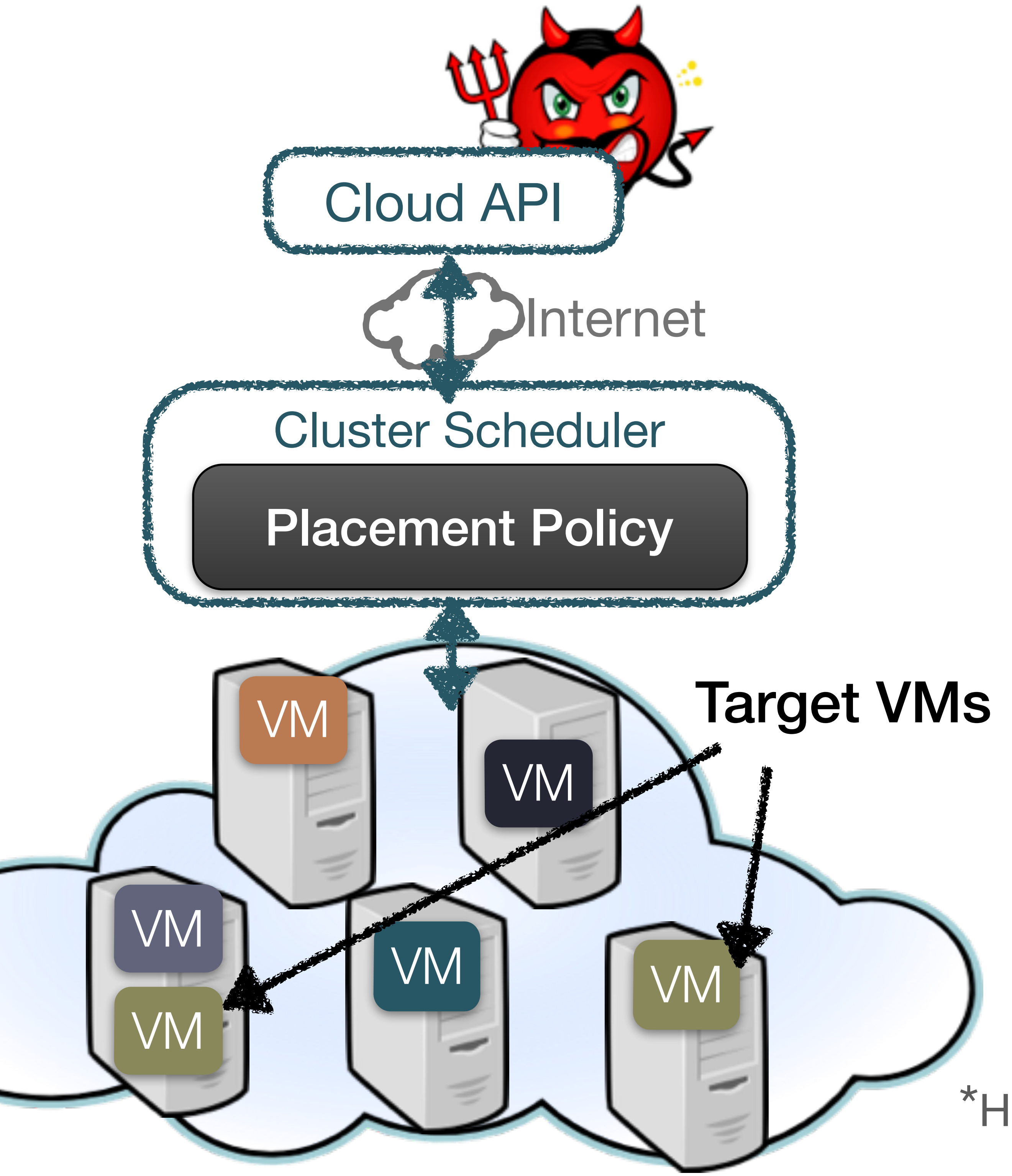
*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study



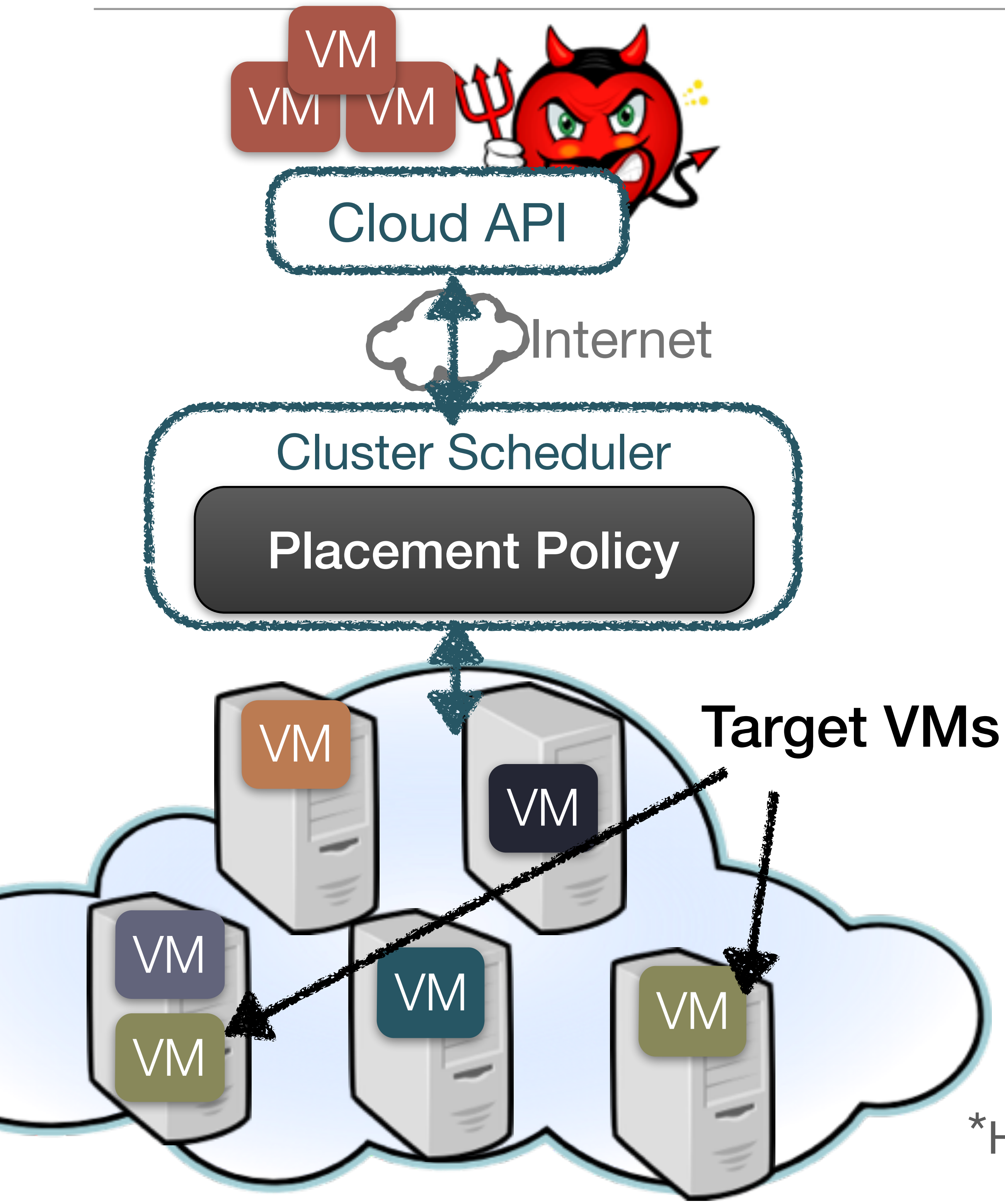
*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study



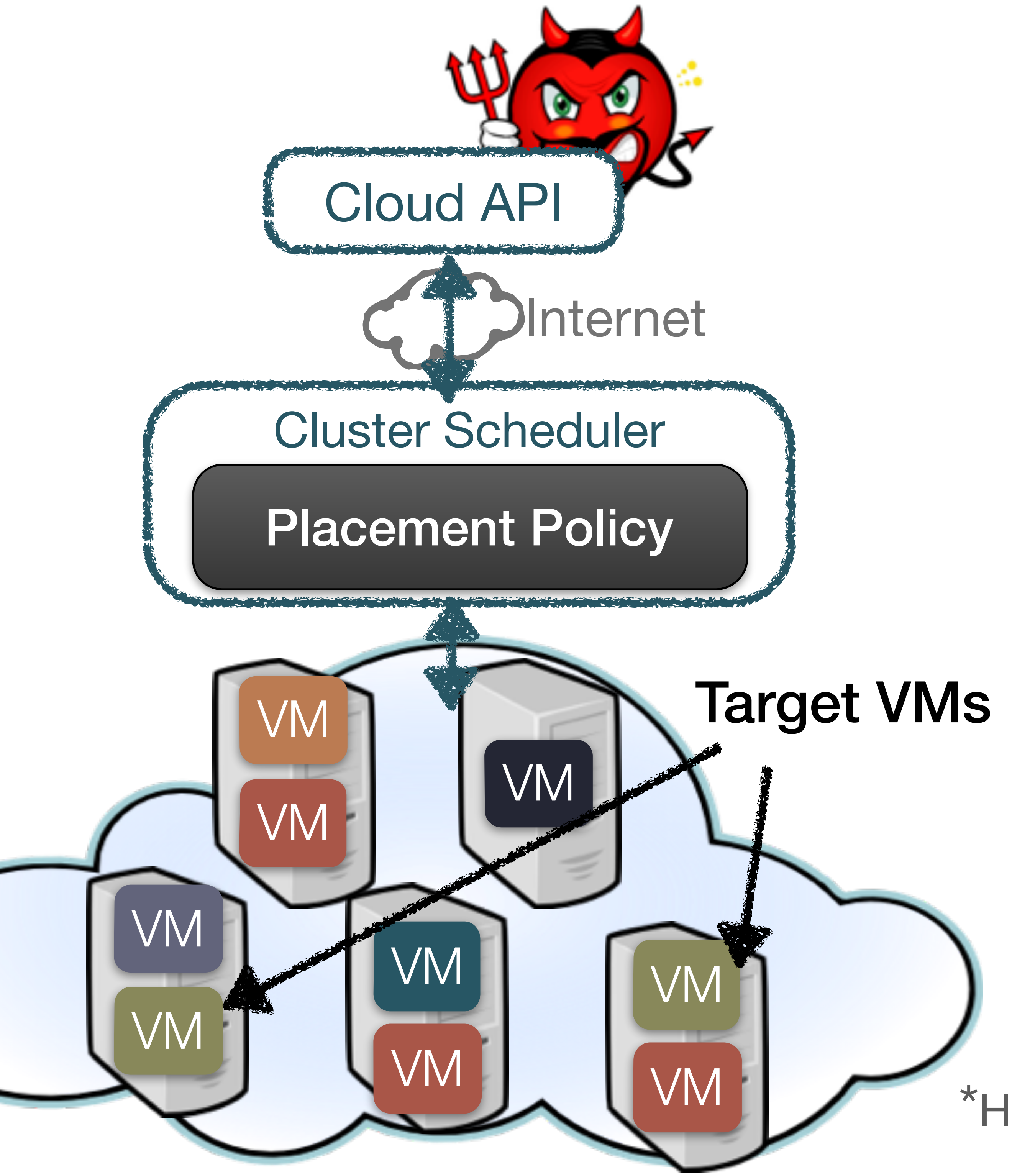
*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study



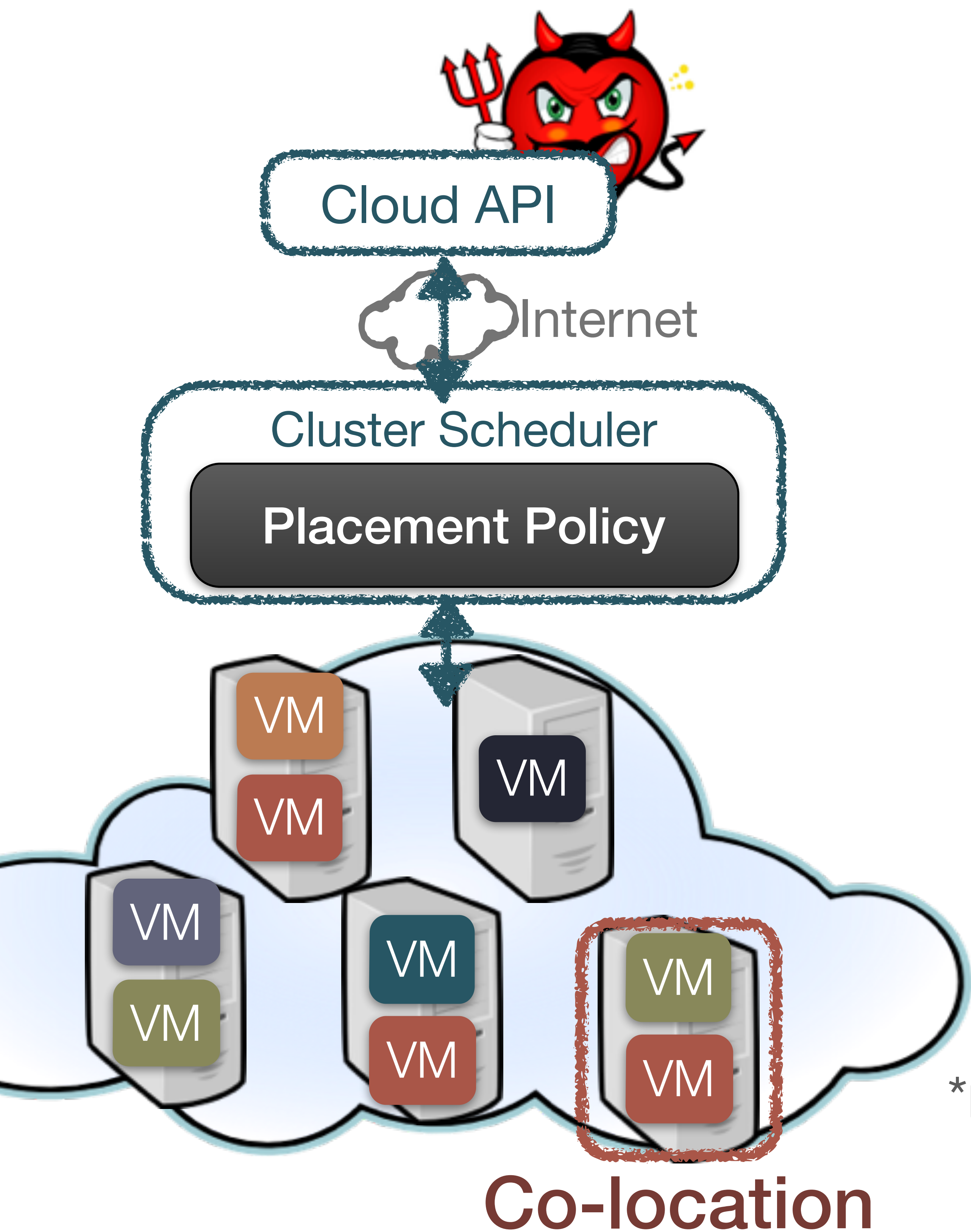
*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study



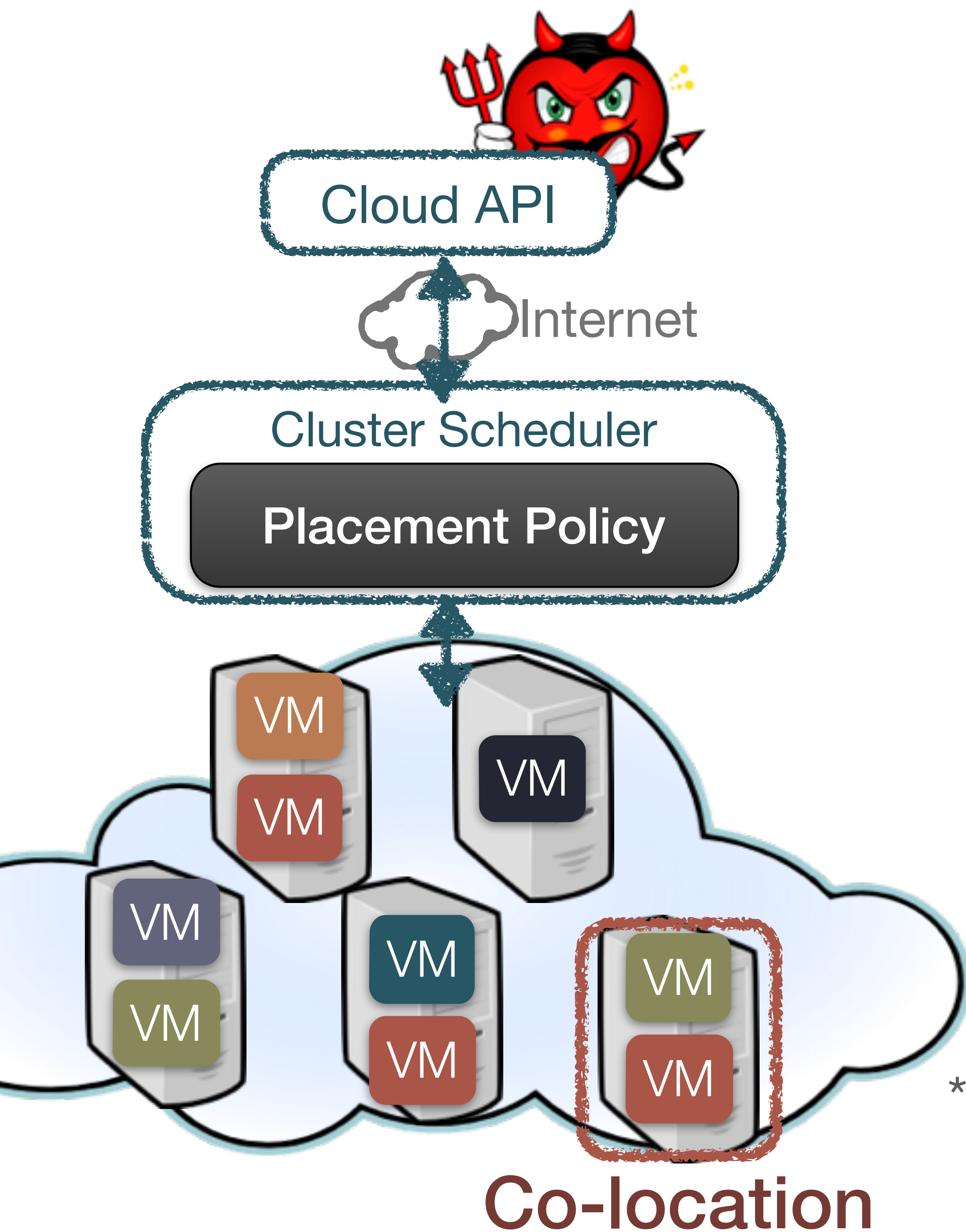
*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study



*Hey you get off my cloud, Ristenpart et al., CCS 2009

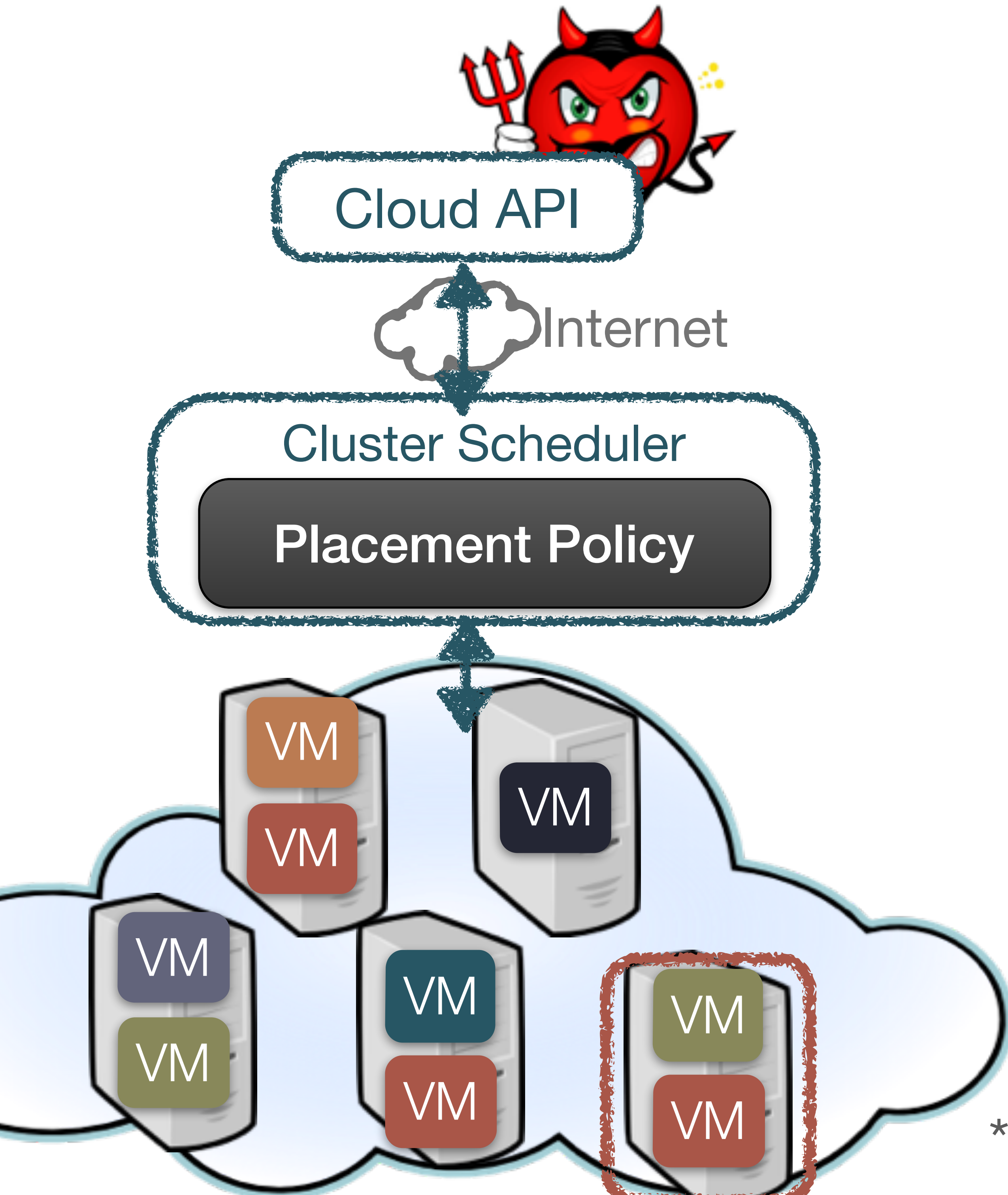
Prior Work on Co-location: 2009 Study



- 6 years old

*Hey you get off my cloud, Ristenpart et al., CCS 2009

Prior Work on Co-location: 2009 Study

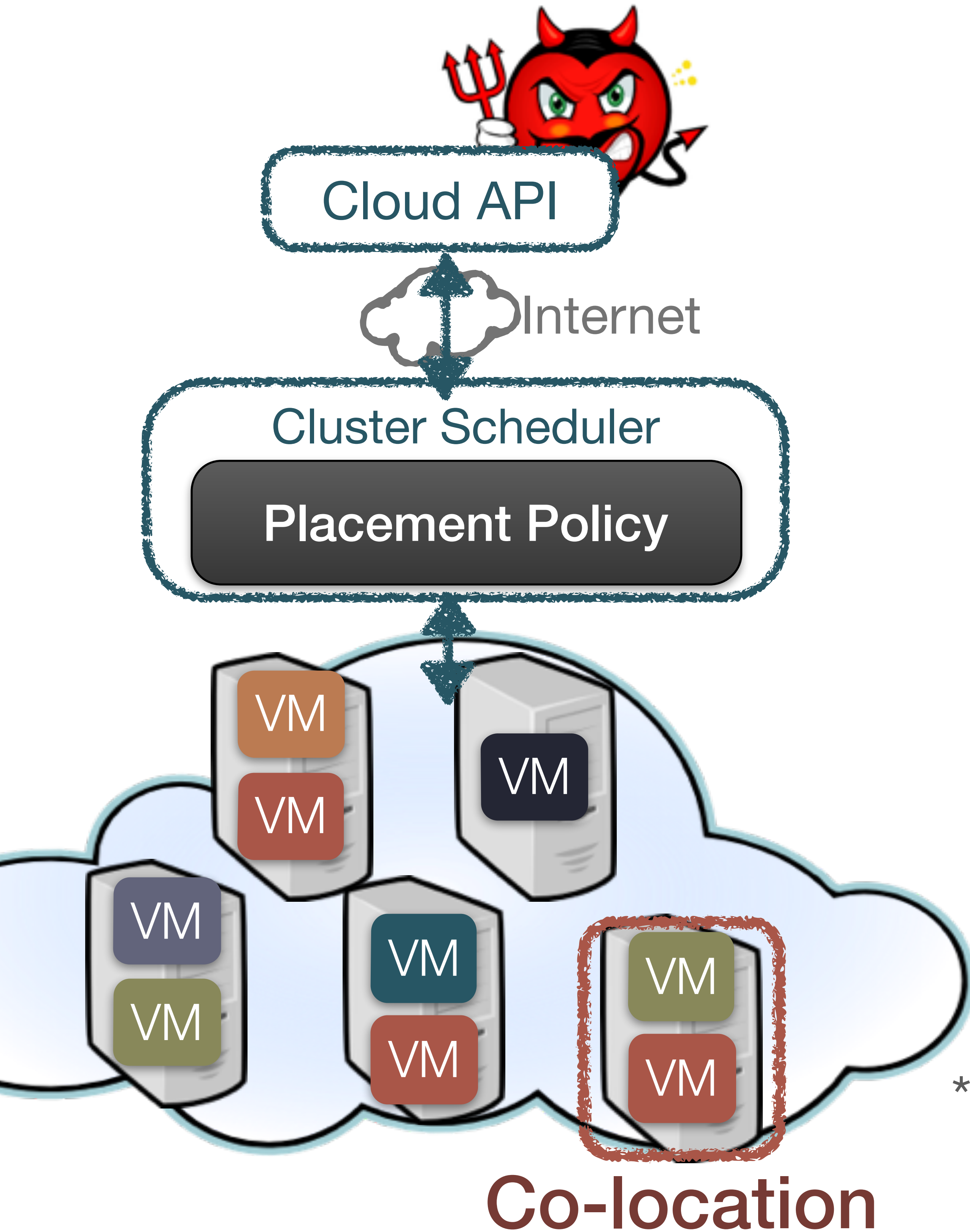


Co-location

- 6 years old
- New countermeasures (e.g, virtual private clouds)

*Hey you get off my cloud, Ristenpart et al., CCS 2009

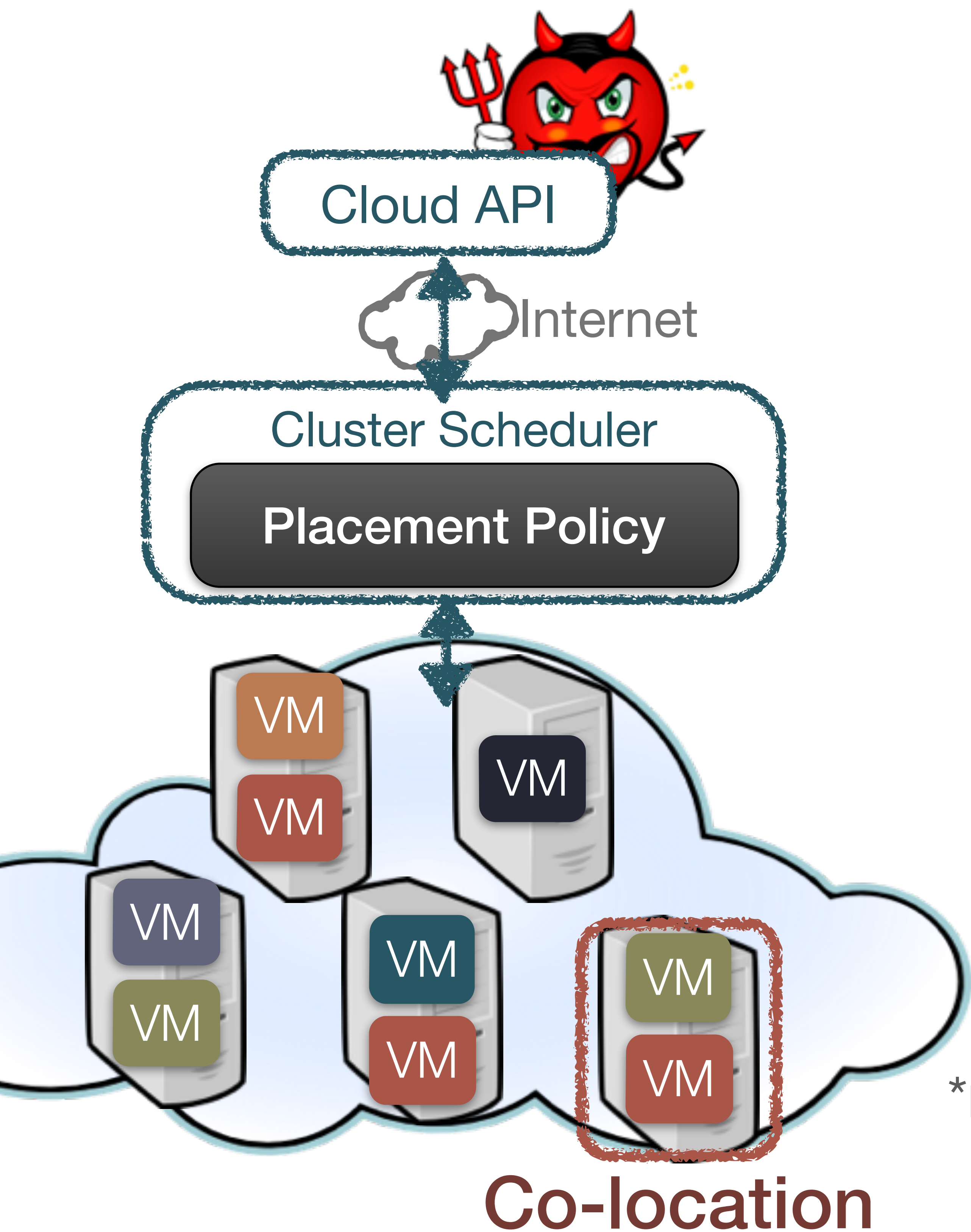
Prior Work on Co-location: 2009 Study



- 6 years old
- New countermeasures (e.g, virtual private clouds)
- Increased scale of clouds

*Hey you get off my cloud, Ristenpart et al., CCS 2009

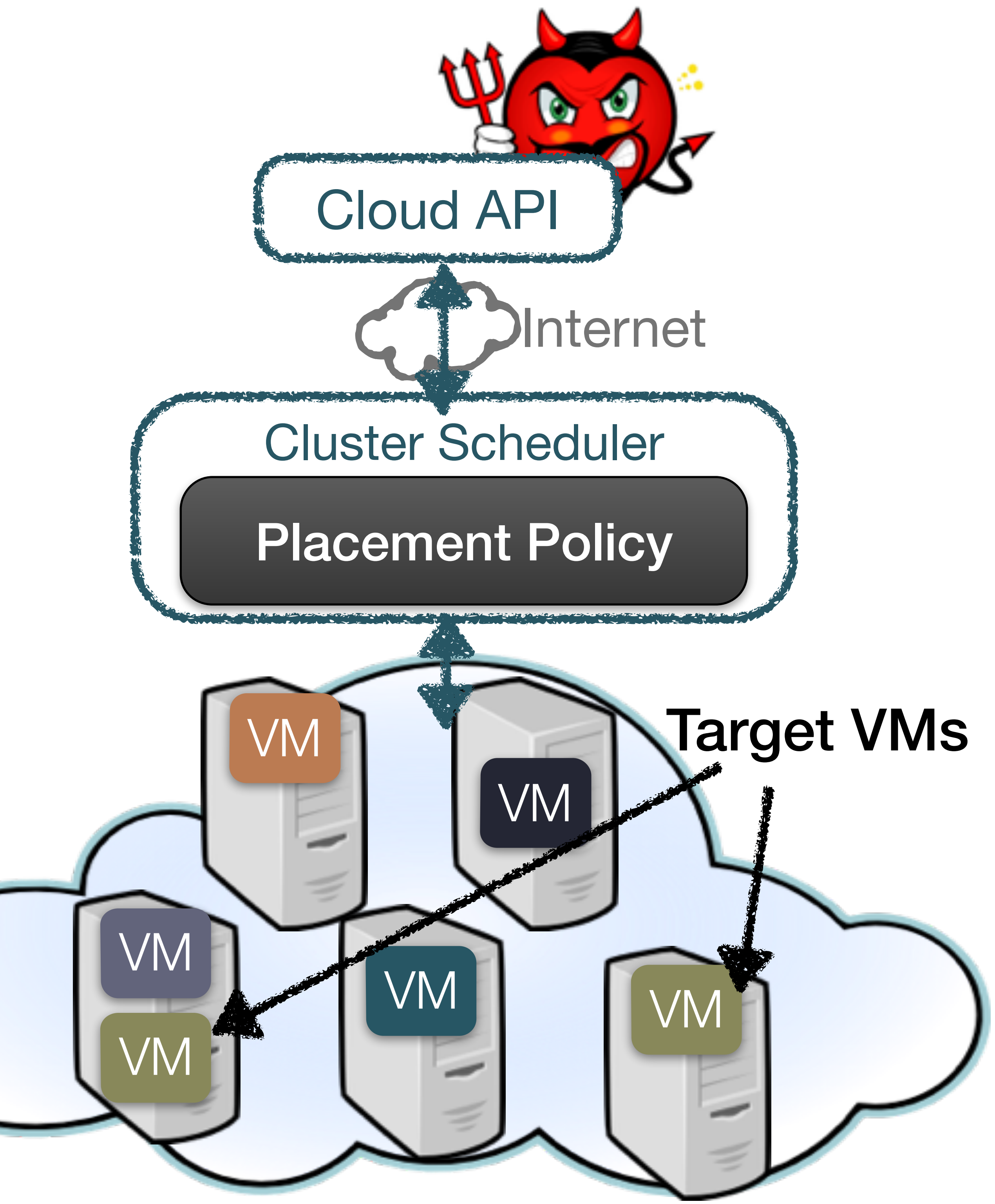
Prior Work on Co-location: 2009 Study



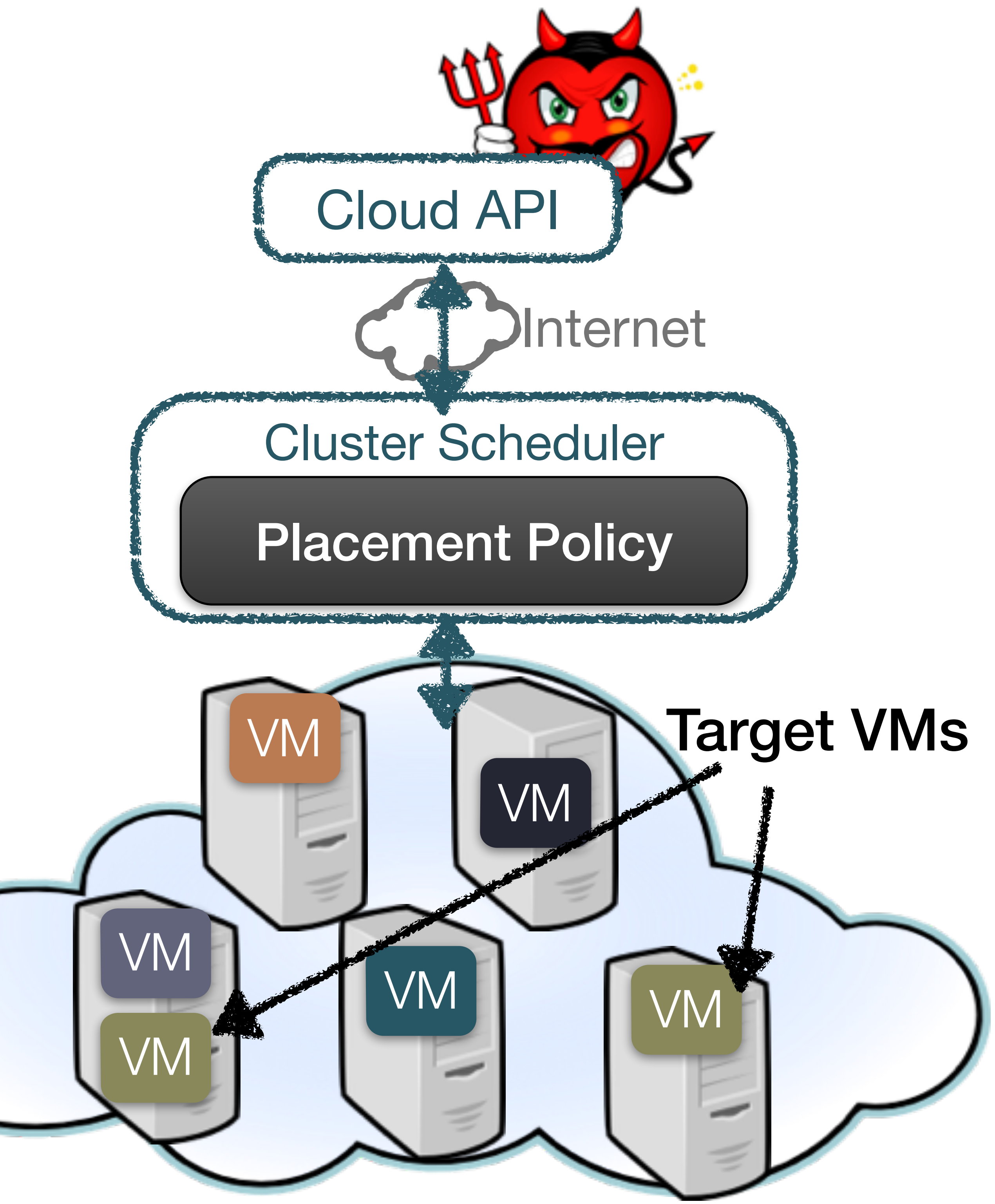
- 6 years old
- New countermeasures (e.g, virtual private clouds)
- Increased scale of clouds
- Only on Amazon EC2

*Hey you get off my cloud, Ristenpart et al., CCS 2009

Our Work:
Exploring Co-location Attacks in Modern Clouds

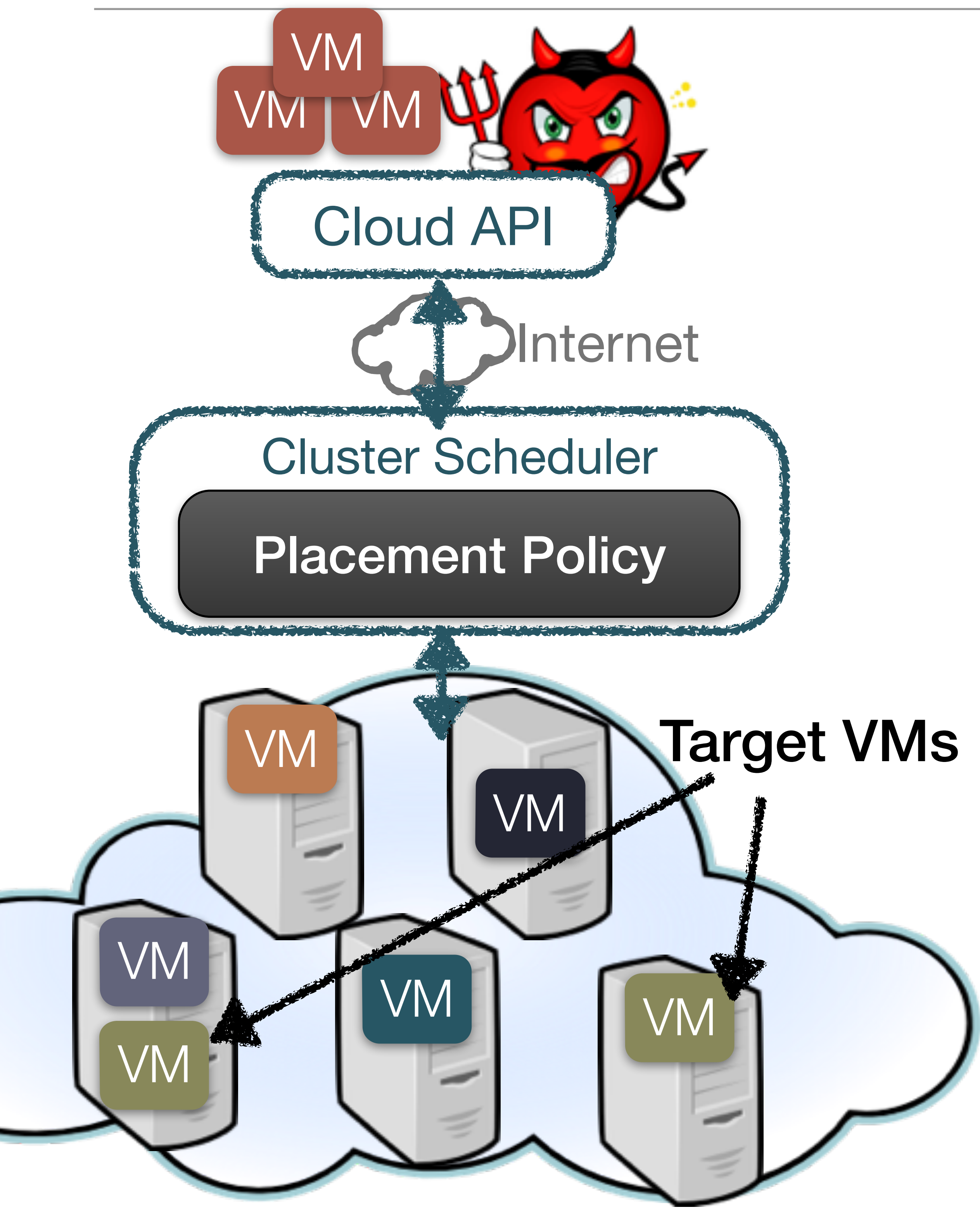


Our Work:
Exploring Co-location Attacks in Modern Clouds



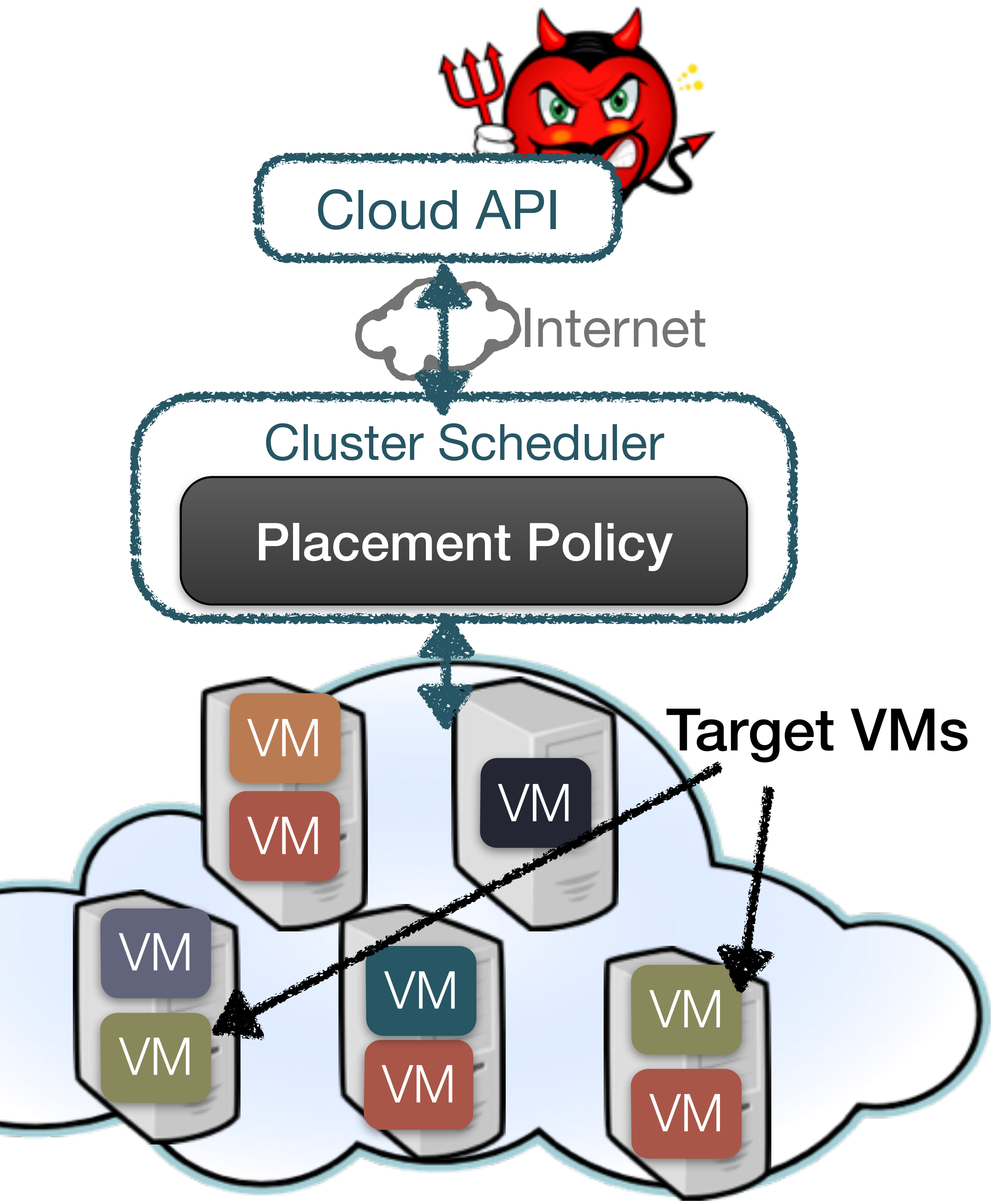
Steps in achieving co-location:

Our Work:
Exploring Co-location Attacks in Modern Clouds



Steps in achieving co-location:

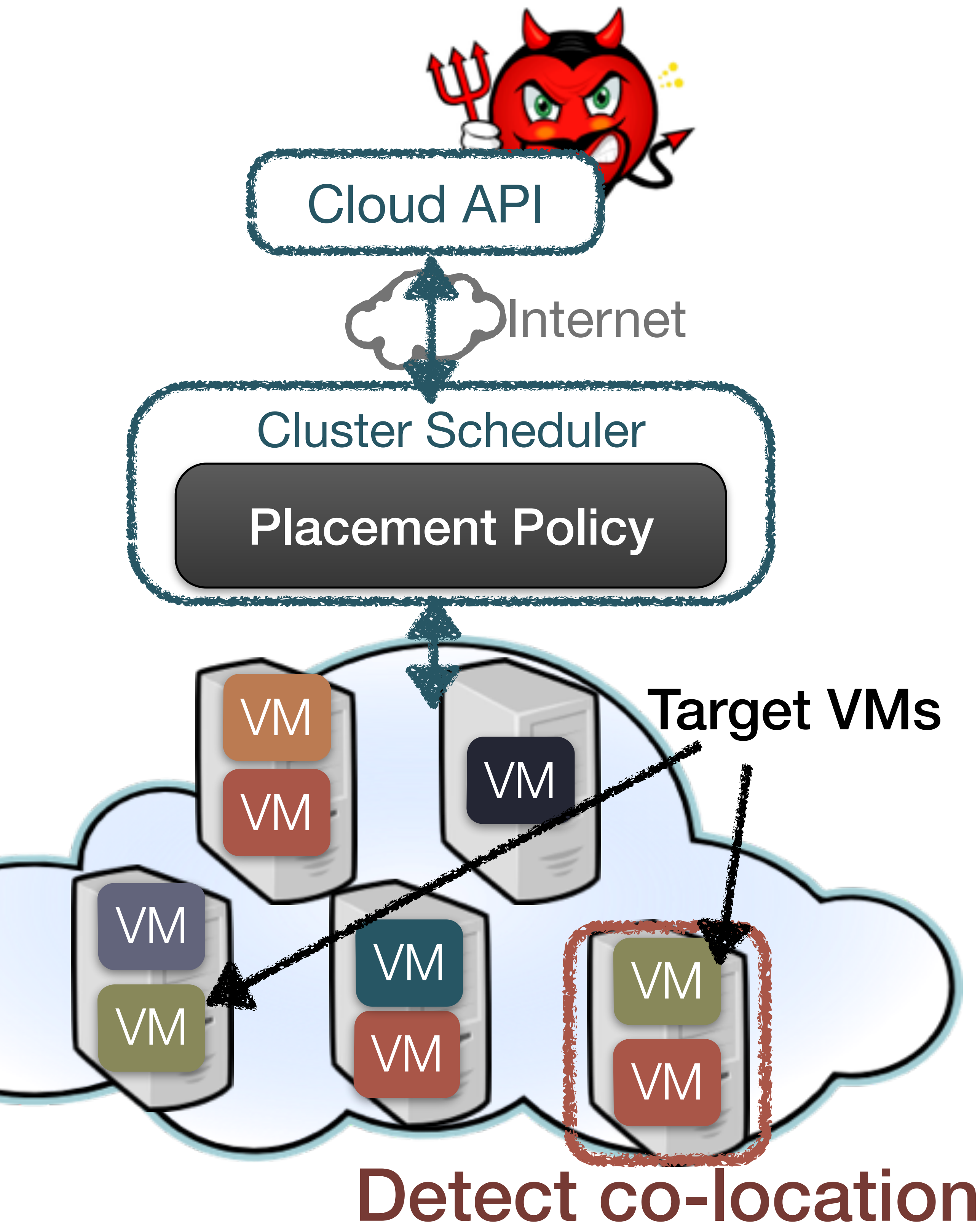
Our Work:
Exploring Co-location Attacks in Modern Clouds



Steps in achieving co-location:

1. Finding Launch Strategy
    - launch parameters to increase chances of co-location

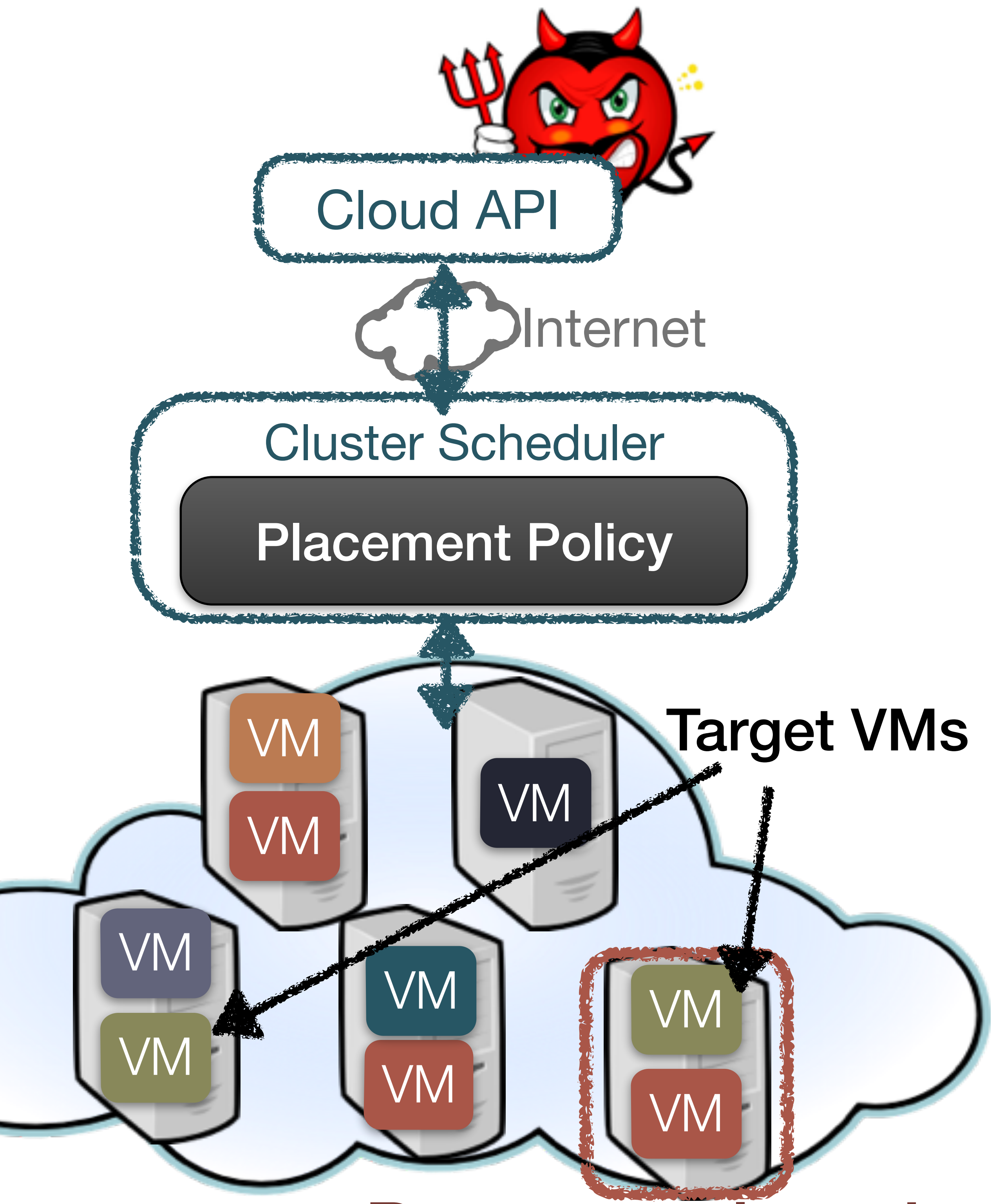
Our Work:
Exploring Co-location Attacks in Modern Clouds



Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location
2. Detecting Co-location
   - with any target victim

Our Work:
Exploring Co-location Attacks in Modern Clouds



Detect co-location

Steps in achieving co-location:

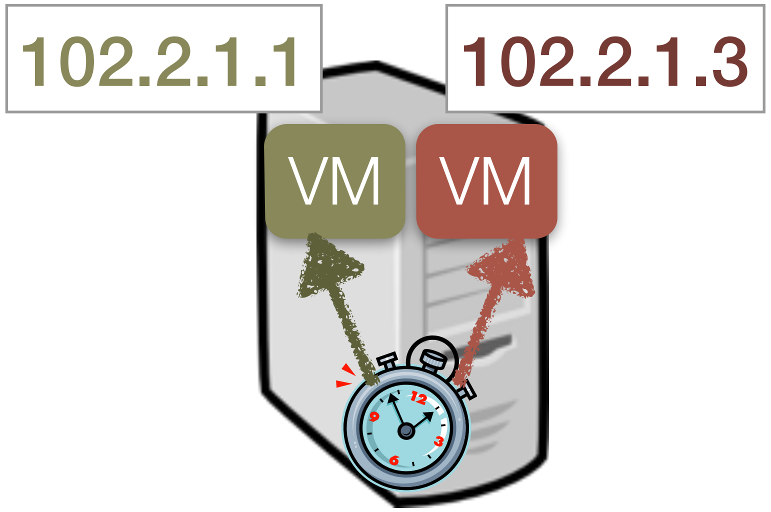
1. Finding Launch Strategy
   - launch parameters to increase chances of co-location
2. Detecting Co-location
   - with any target victim

Co-residency Detection

Co-residency Detection

1. Read shared state
   on two VMs

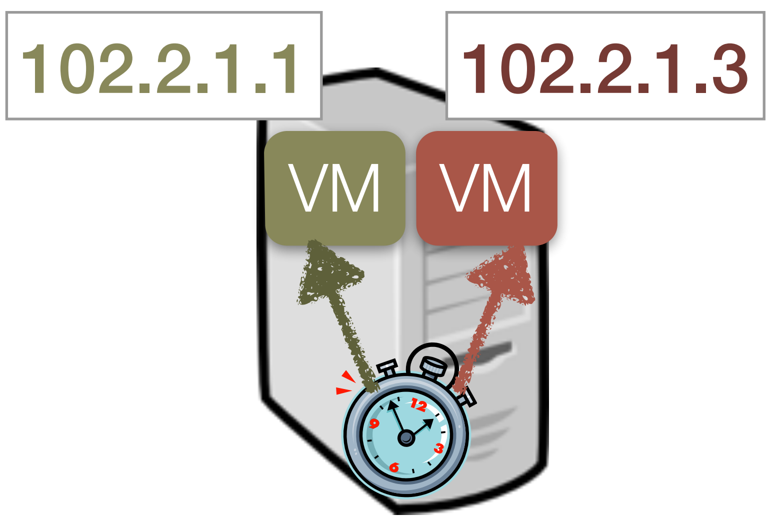
   e.g., private IP addresses,
   shared TSC counters.



Co-residency Detection

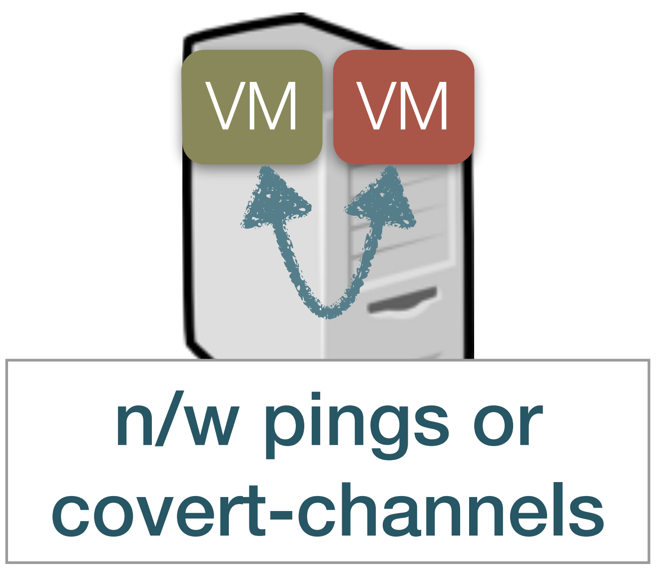
1. Read shared state on two VMs

e.g., private IP addresses, shared TSC counters.



2. Correlate performance of shared resources

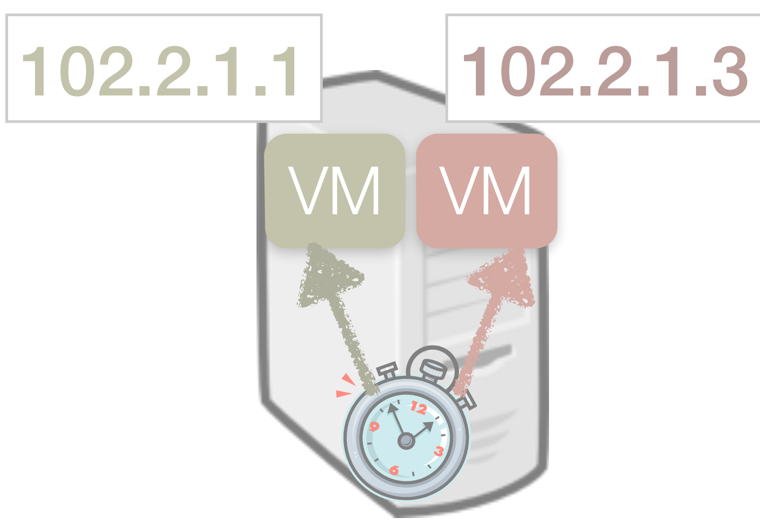
e.g., network round-trip times, cache-based covert-channels.



Co-residency Detection

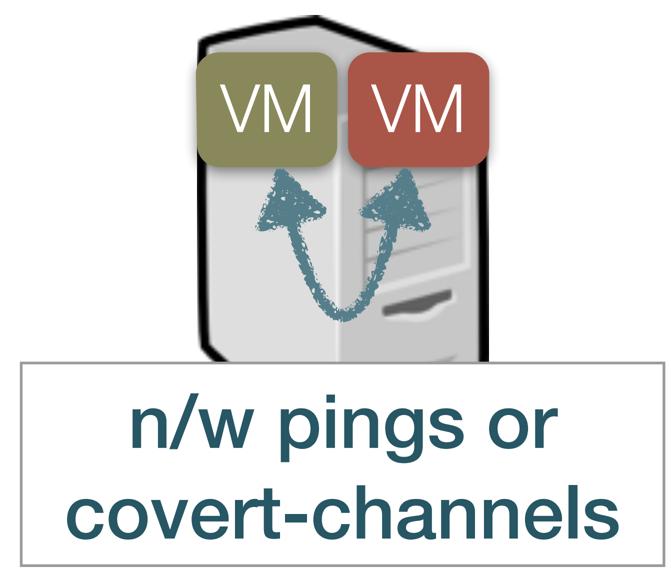
1. Read shared state on two VMs

e.g., private IP addresses, shared TSC counters.



2. Correlate performance of shared resources

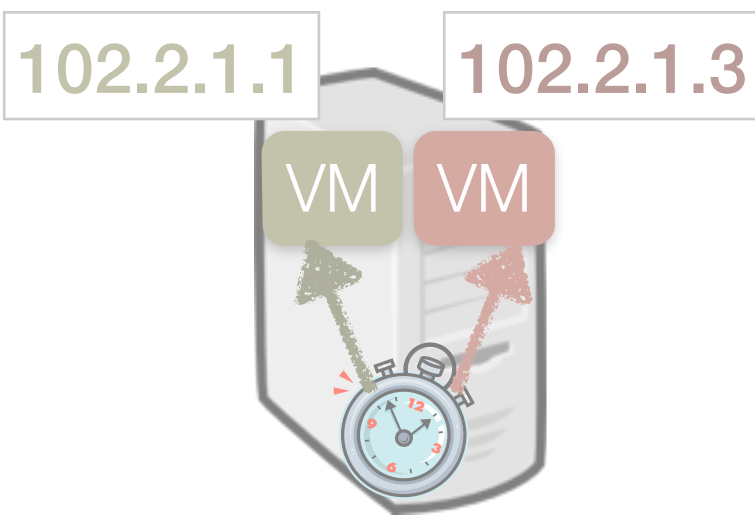
e.g., network round-trip times, cache-based covert-channels.



Co-residency Detection

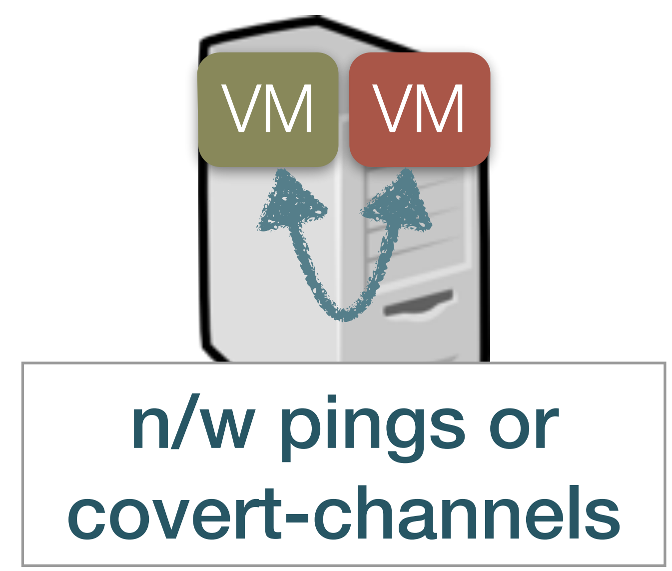
1. Read shared state on two VMs

e.g., private IP addresses, shared TSC counters.



2. Correlate performance of shared resources

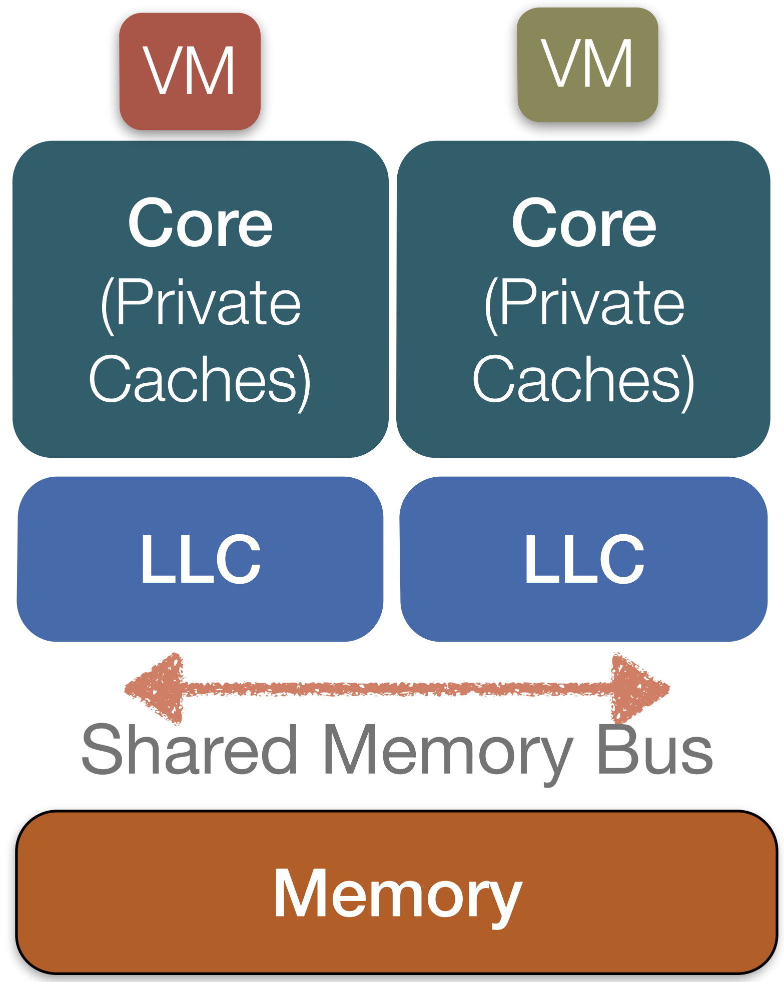
e.g., network round-trip times, cache-based covert-channels.



A memory-based covert-channel* can cause 3x-4x degradation

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12

Background: Memory Covert-Channel

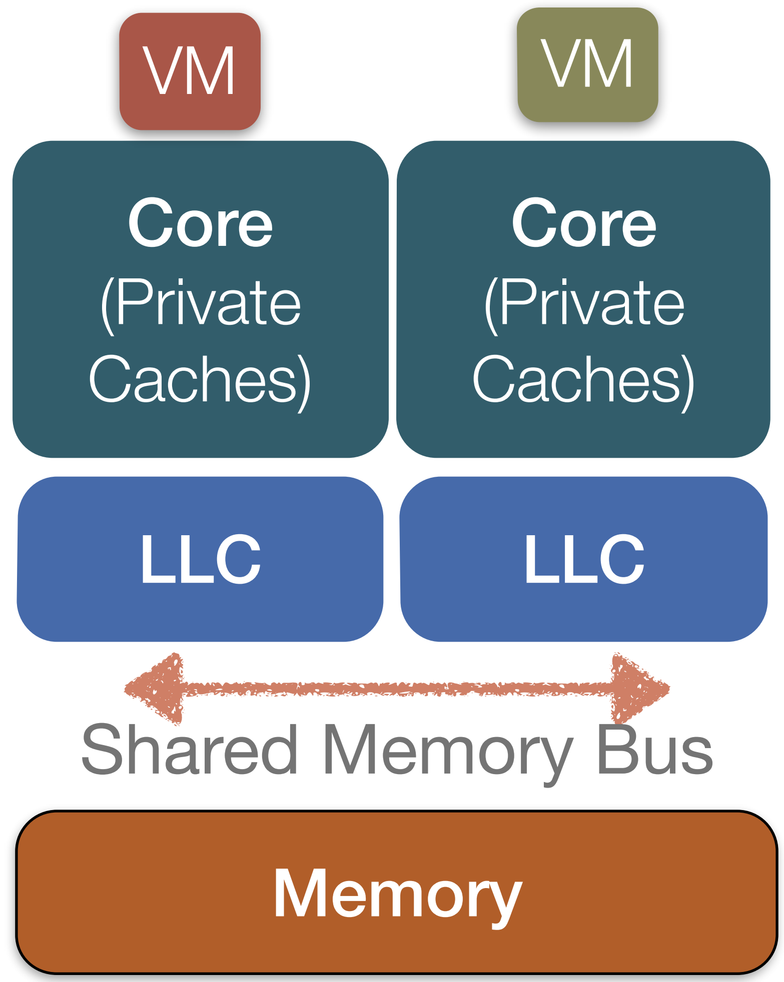


* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```

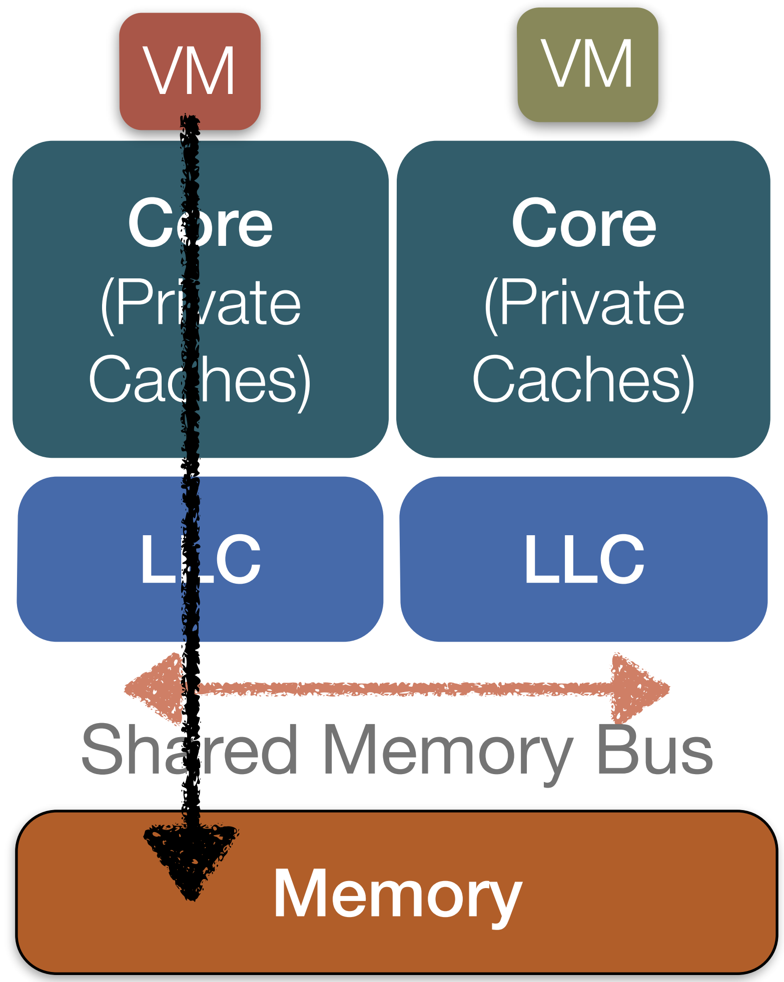


* Wu et al. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.", Usenix Security'12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```

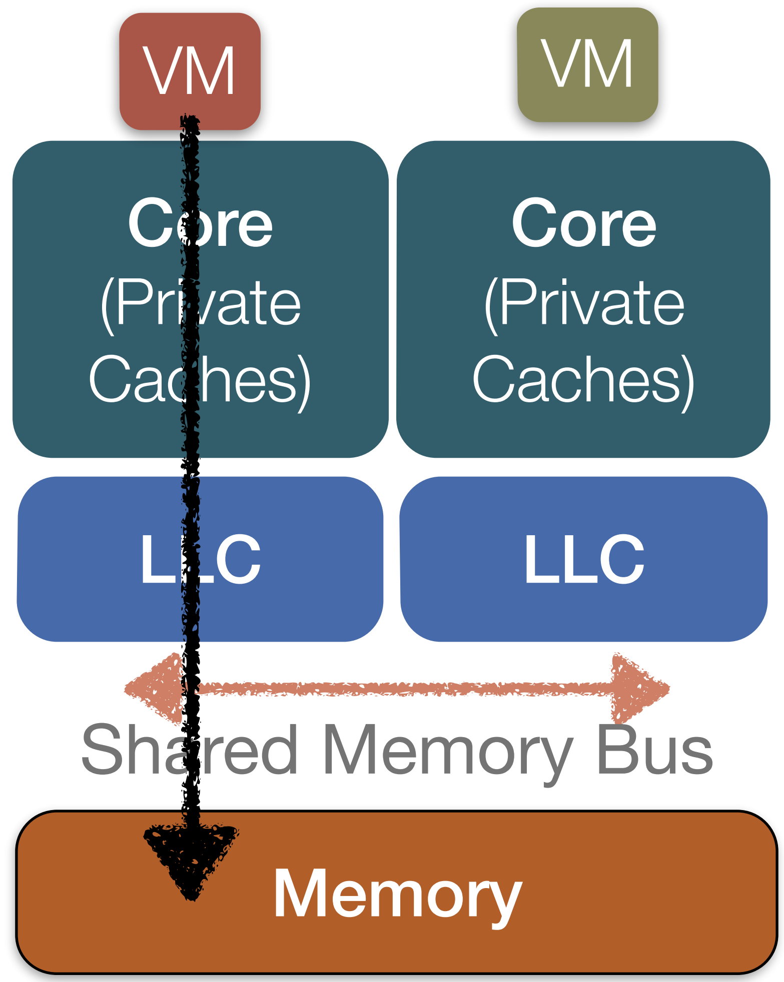


* Wu et al. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.", Usenix Security'12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```



Receiver:

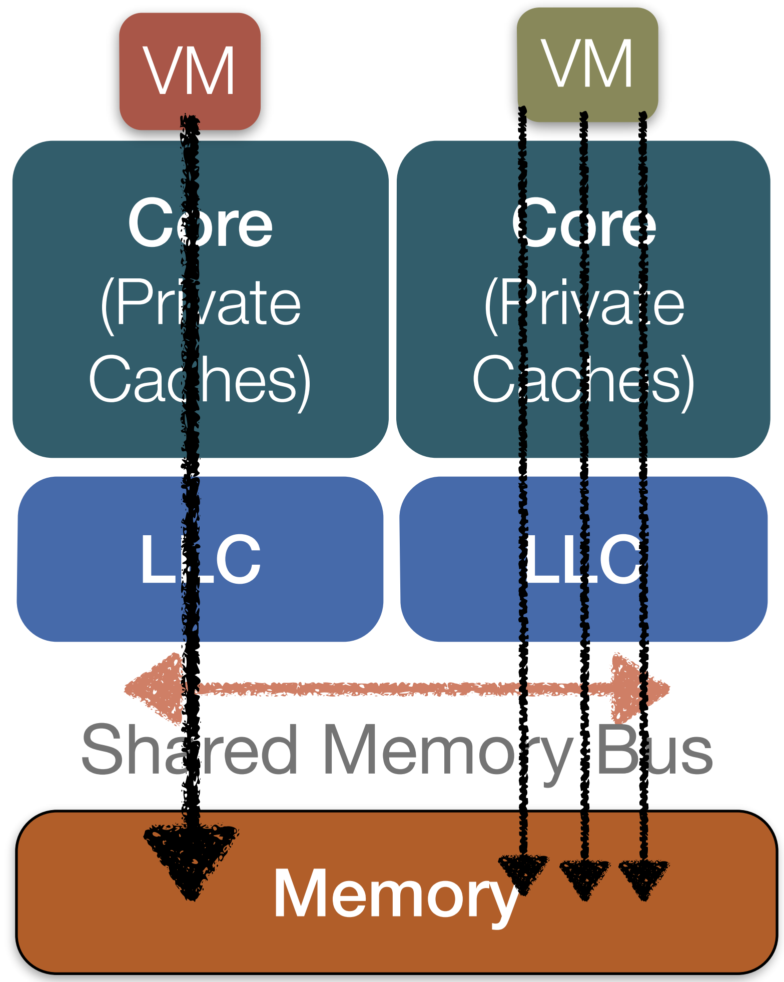
```
Observe() {
  s = start_time
  repeat N
    mem_access()
  done
  e = end_time
  bw = N/(e-s)
}
```

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```



Receiver:

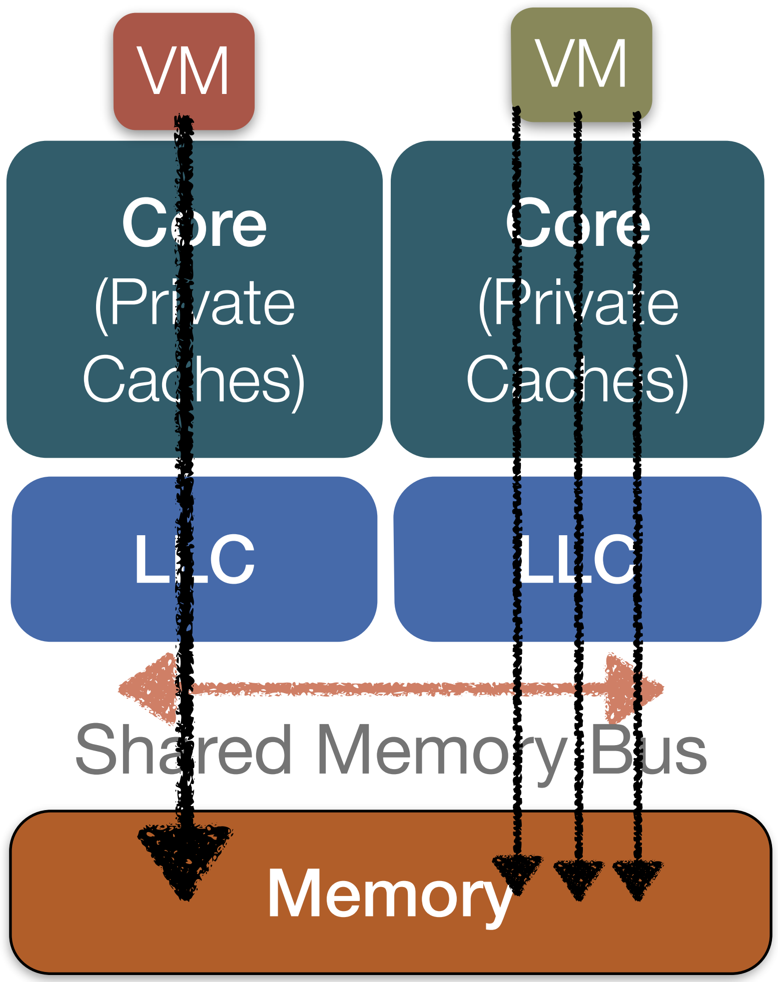
```
Observe() {
  s = start_time
  repeat N
    mem_access()
  done
  e = end_time
  bw = N/(e-s)
}
```

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```



Receiver:

```
Observe() {
  s = start_time
  repeat N
    mem_access()
  done
  e = end_time
  bw = N/(e-s)
}
```

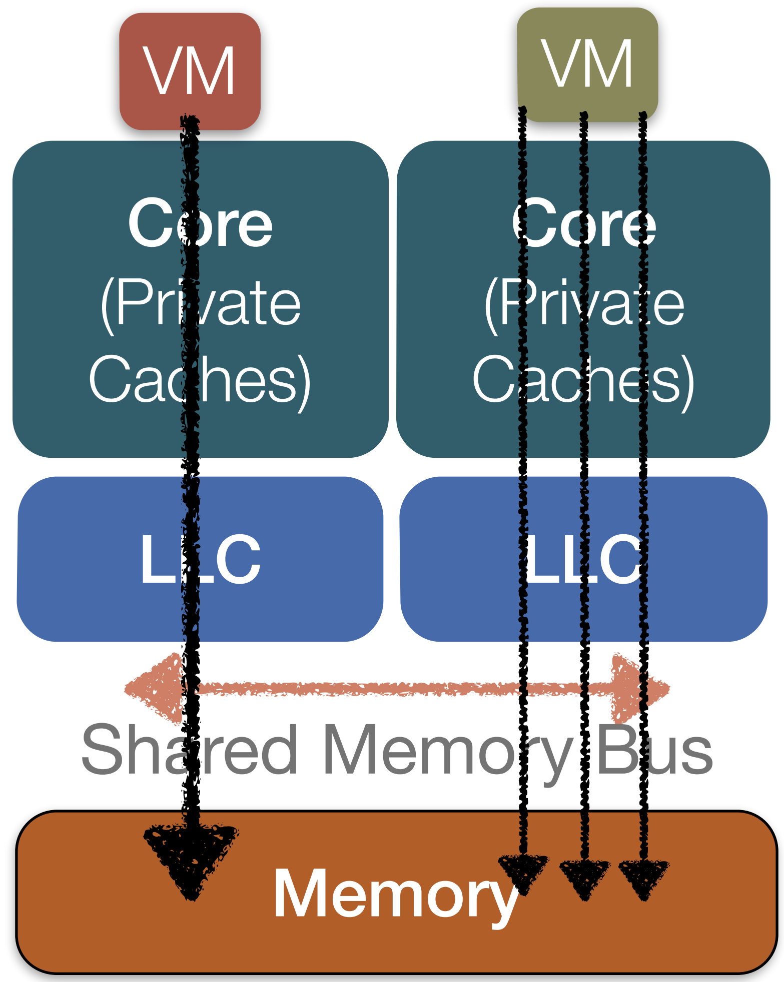
"Victim" VM must cooperate with attack VM
O.K. for measurement studies

* Wu et al. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.", Usenix Security'12

Background: Memory Covert-Channel

Sender:

```
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```



Receiver:

```
Observe() {
  s = start_time
  repeat N
    mem_access()
  done
  e = end_time
  bw = N/(e-s)
}
```

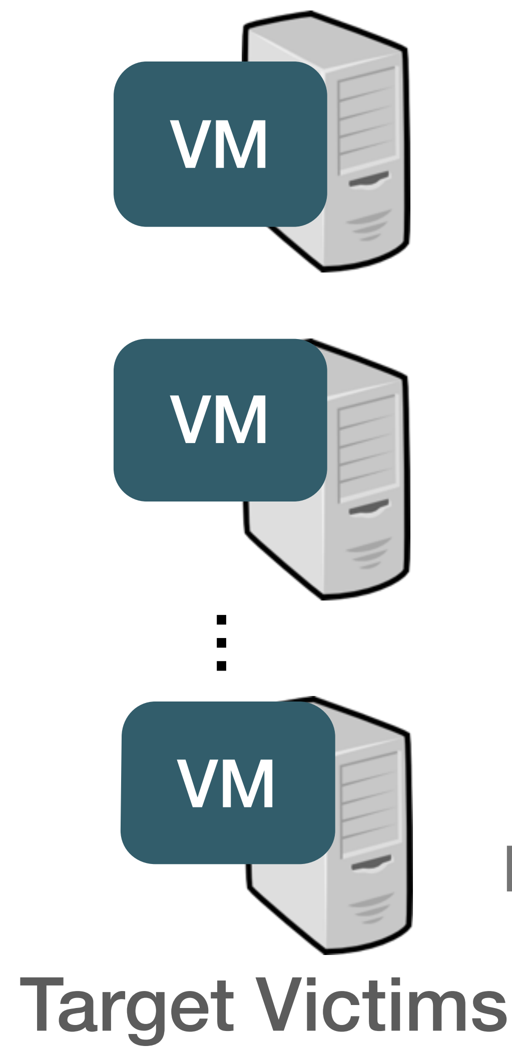
"Victim" VM must cooperate with attack VM
O.K. for measurement studies
But not useful for real attacks in the wild

* Wu et al. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.", Usenix Security'12

Co-residency Detection on Uncooperative Victim

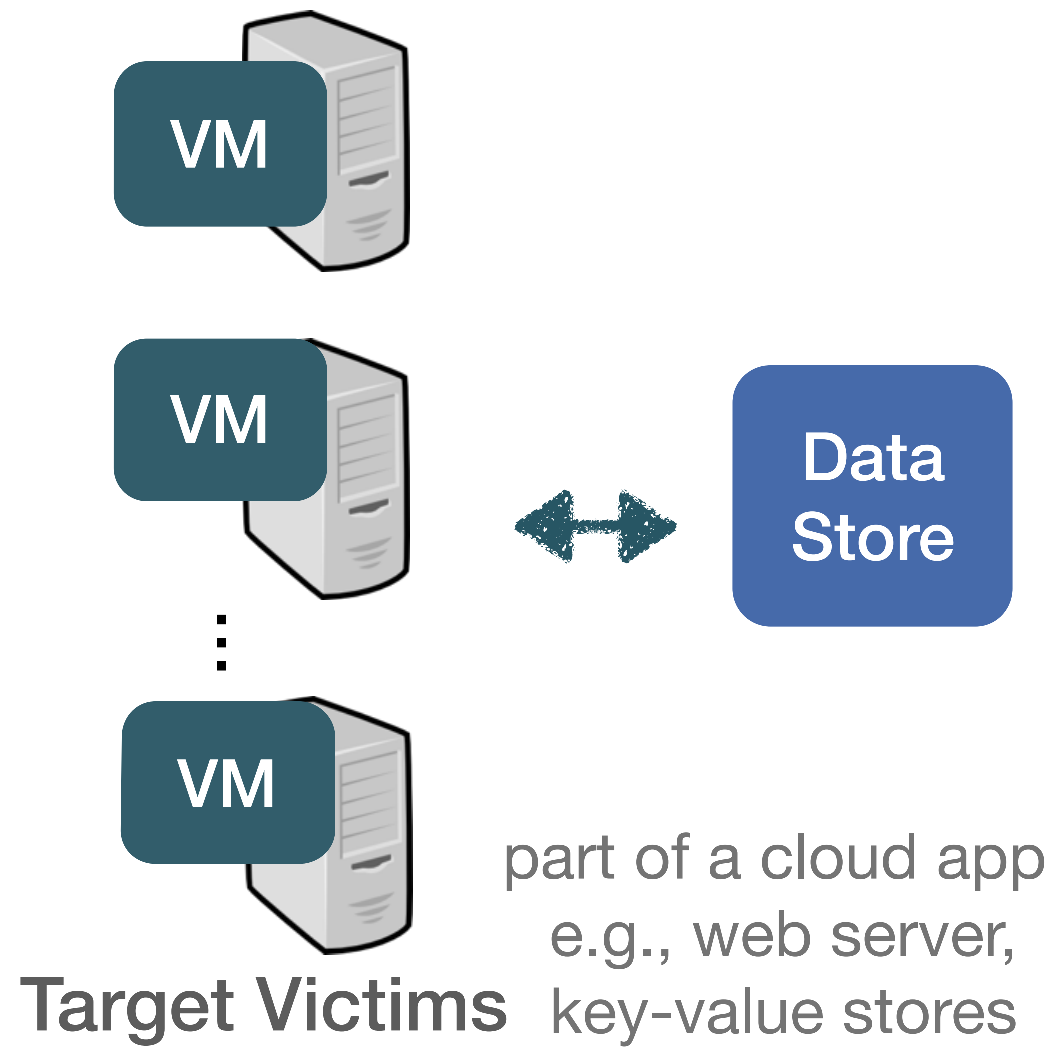
Co-residency Detection on Uncooperative Victim

Co-residency Detection on Uncooperative Victim

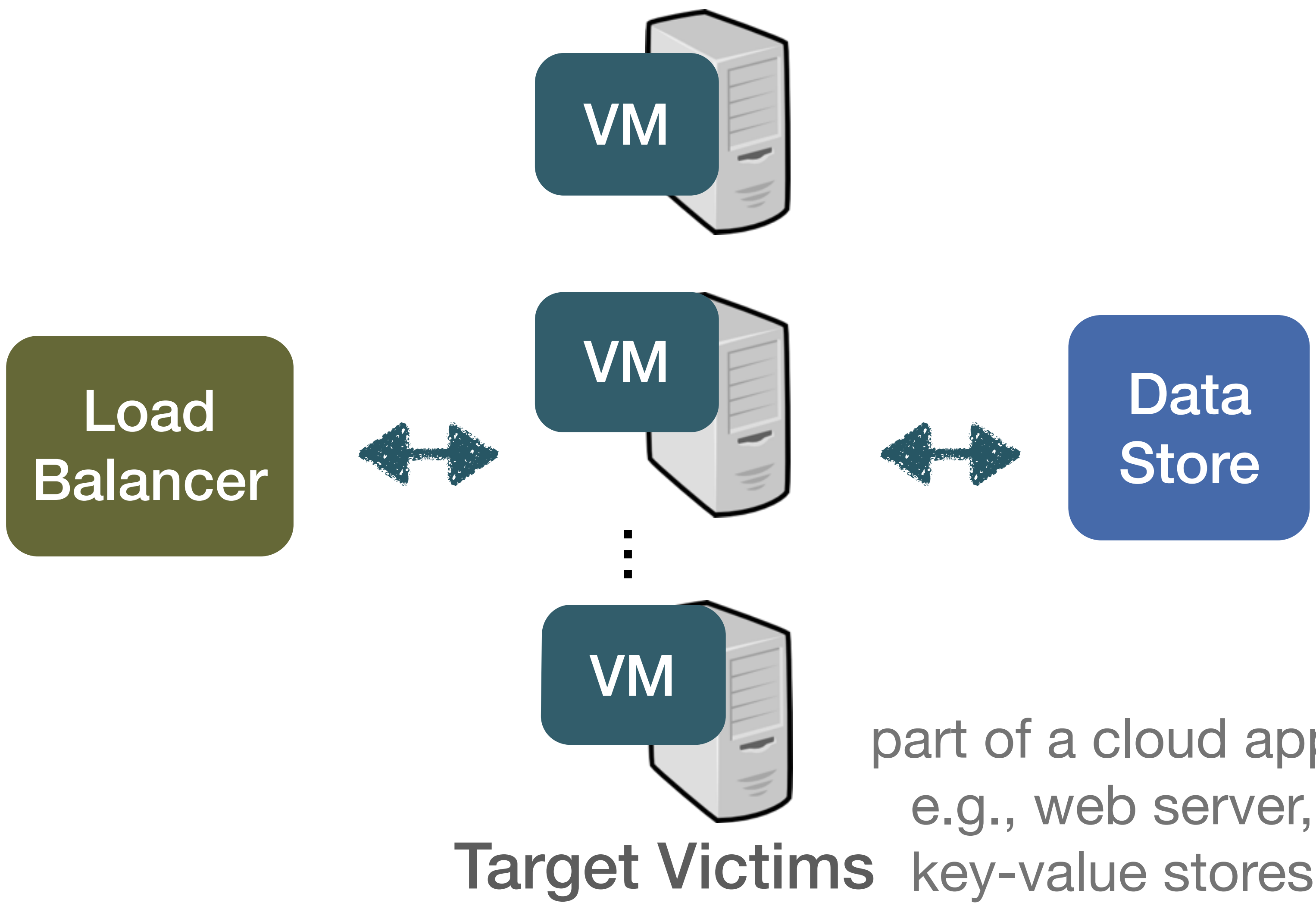


part of a cloud app,
e.g., web server,
key-value stores

Co-residency Detection on Uncooperative Victim

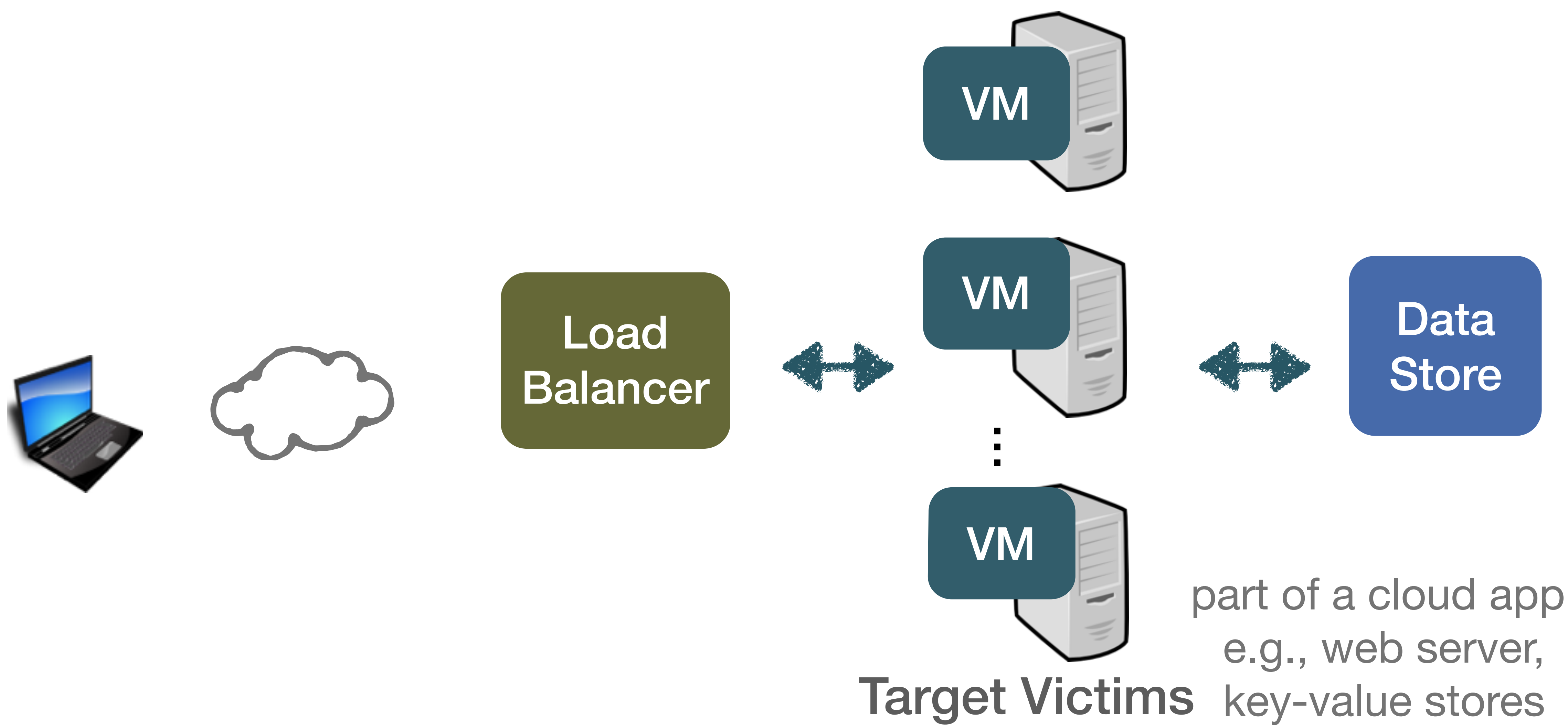


Co-residency Detection on Uncooperative Victim

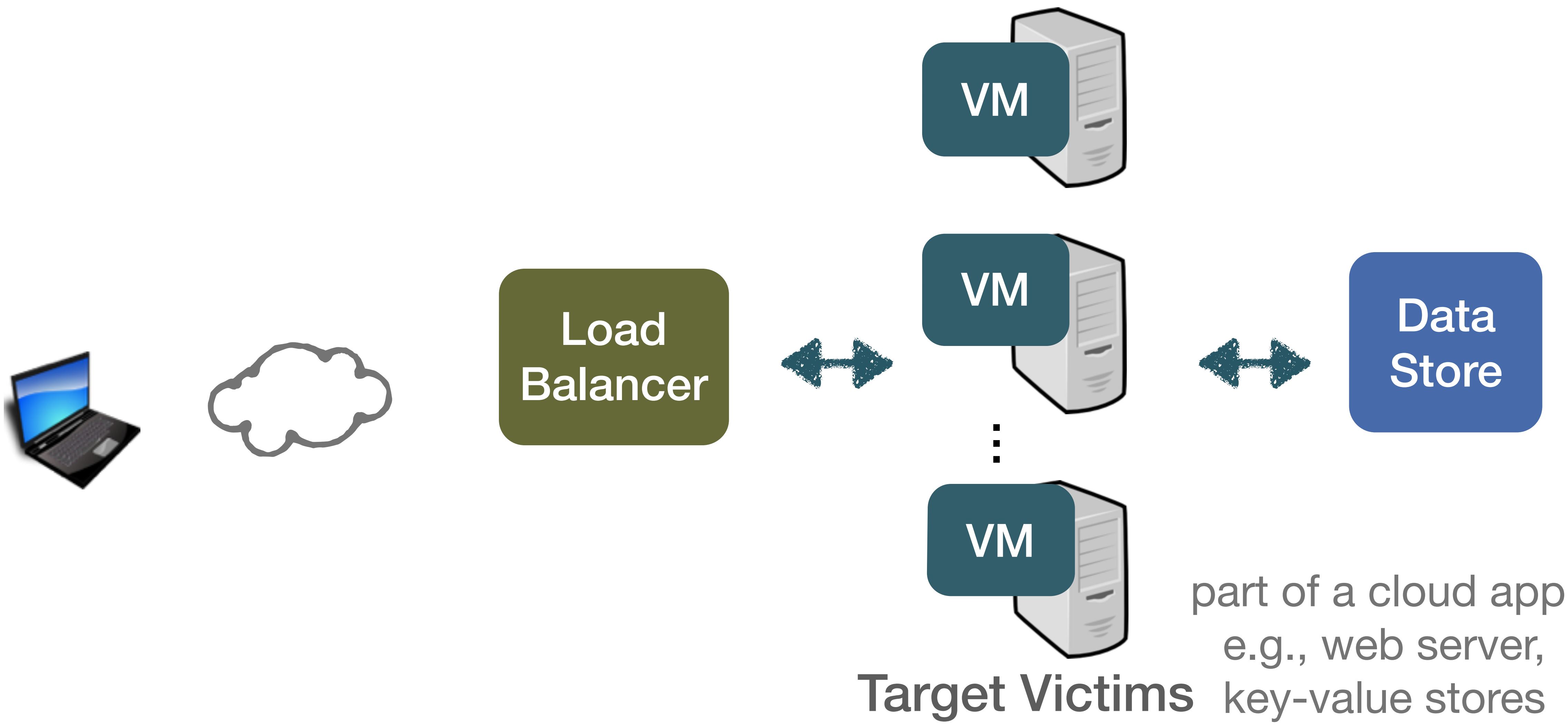


part of a cloud app,
e.g., web server,
key-value stores

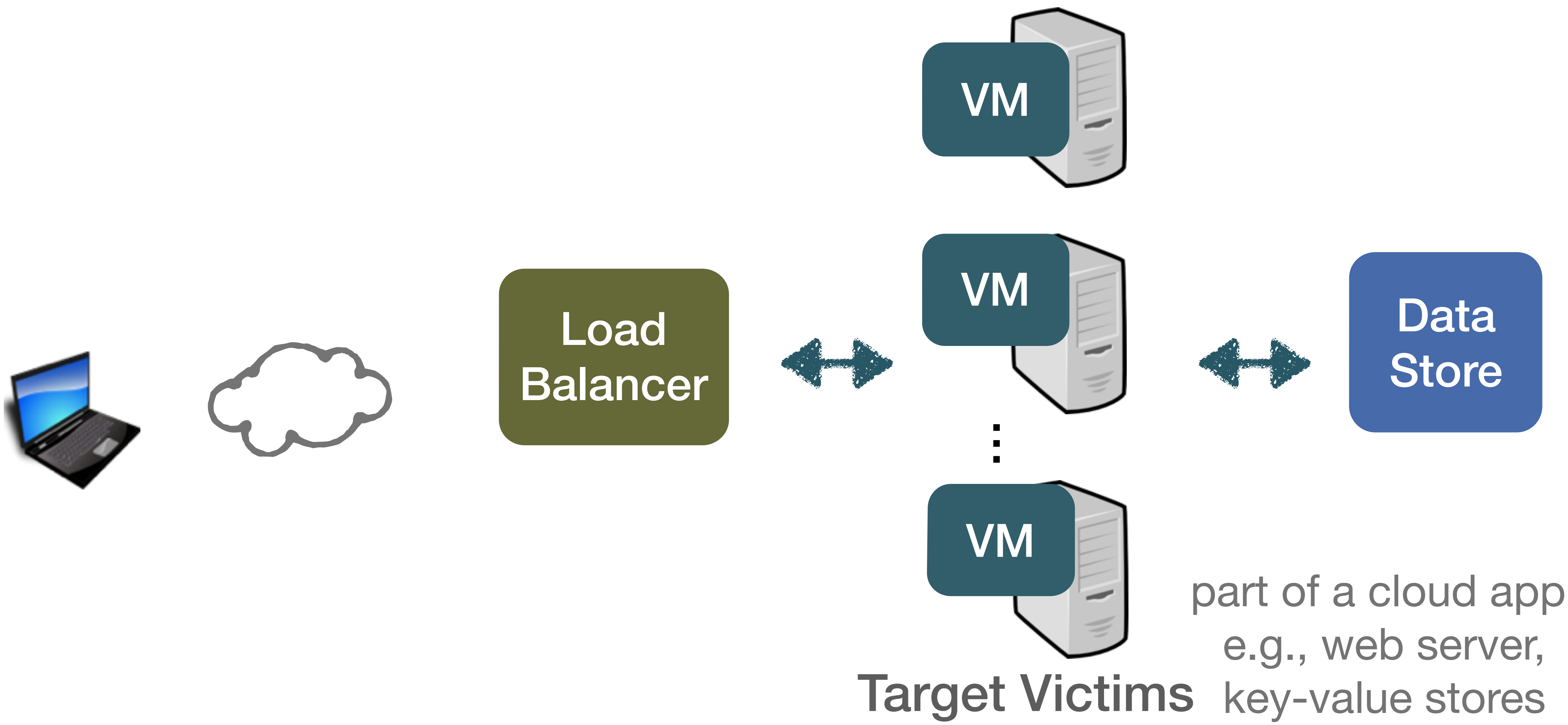
Co-residency Detection on Uncooperative Victim



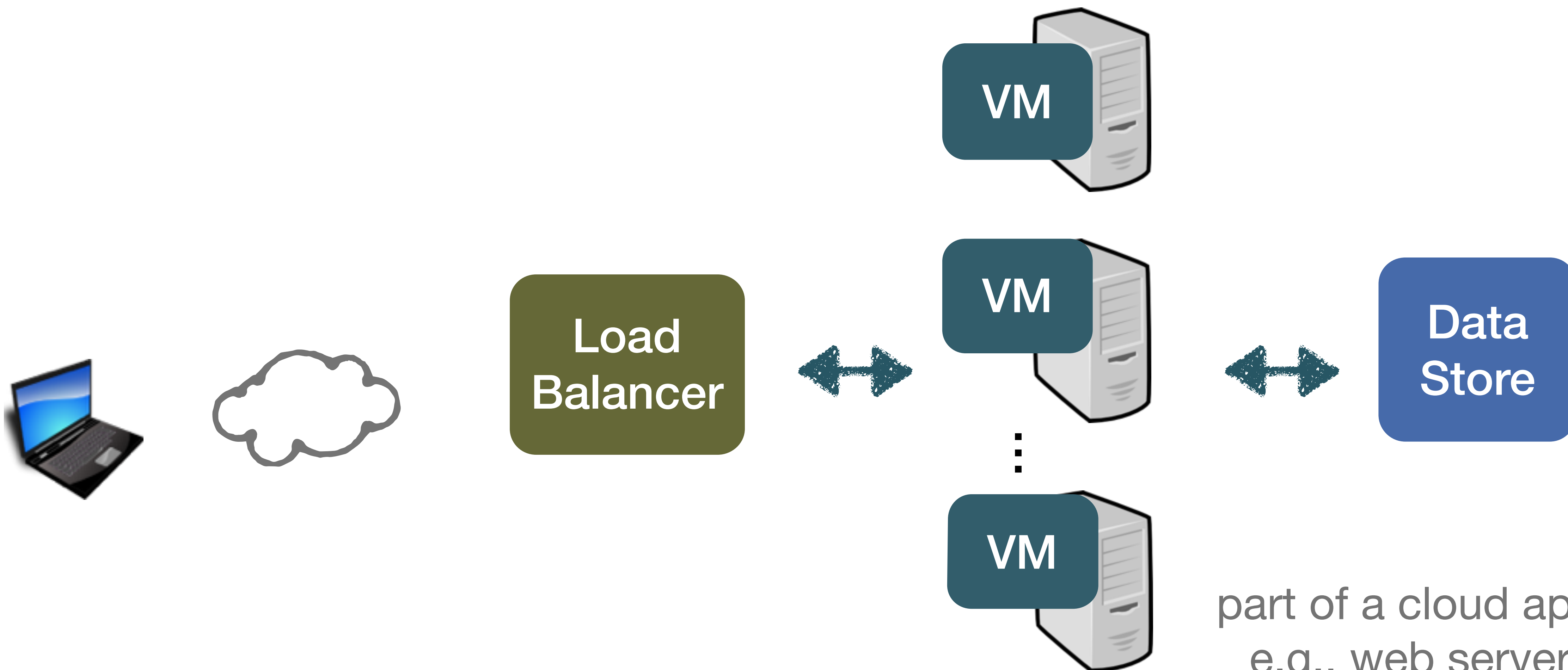
Co-residency Detection on Uncooperative Victim



Co-residency Detection on Uncooperative Victim



Co-residency Detection on Uncooperative Victim

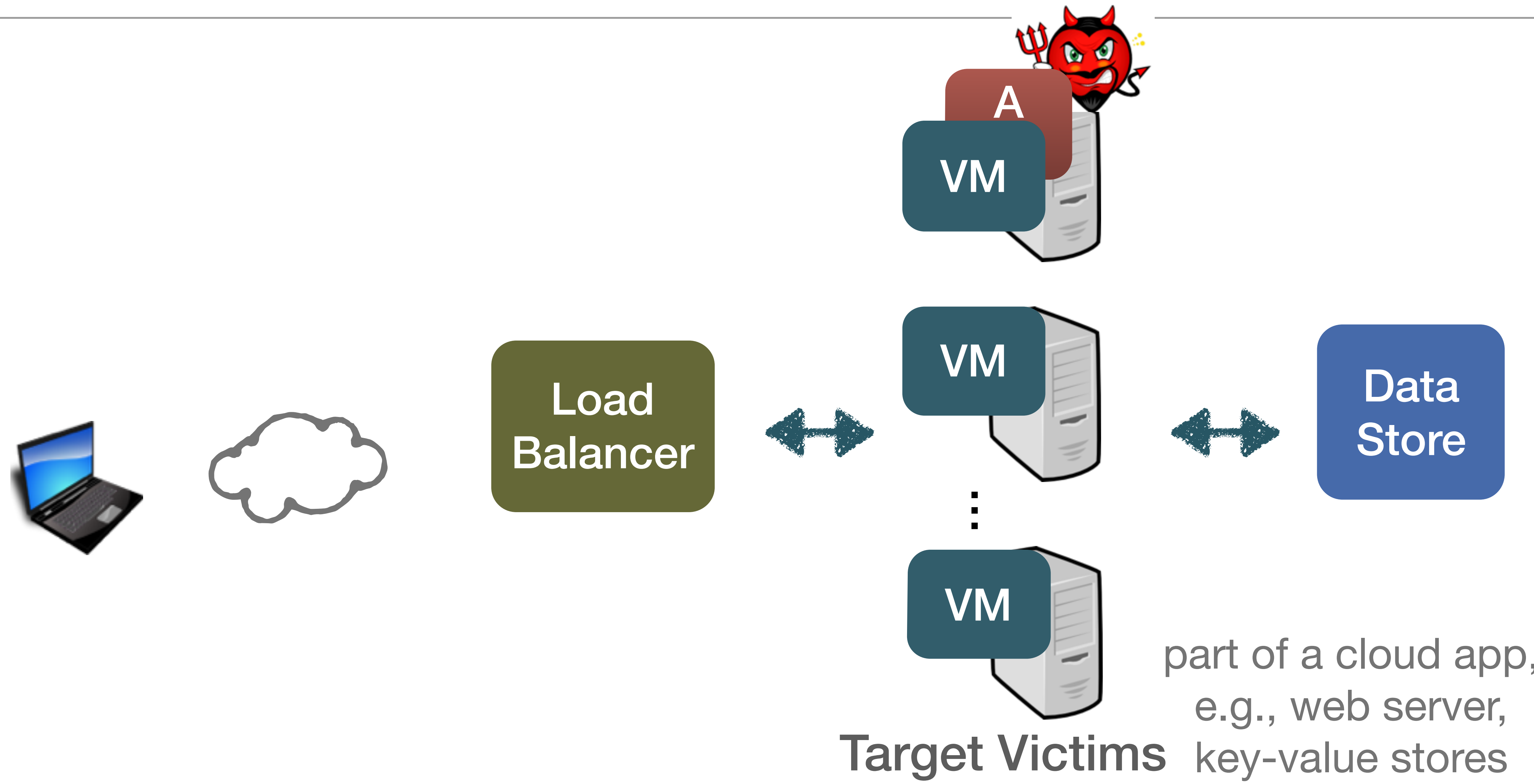


Target Victims

part of a cloud app, e.g., web server, key-value stores

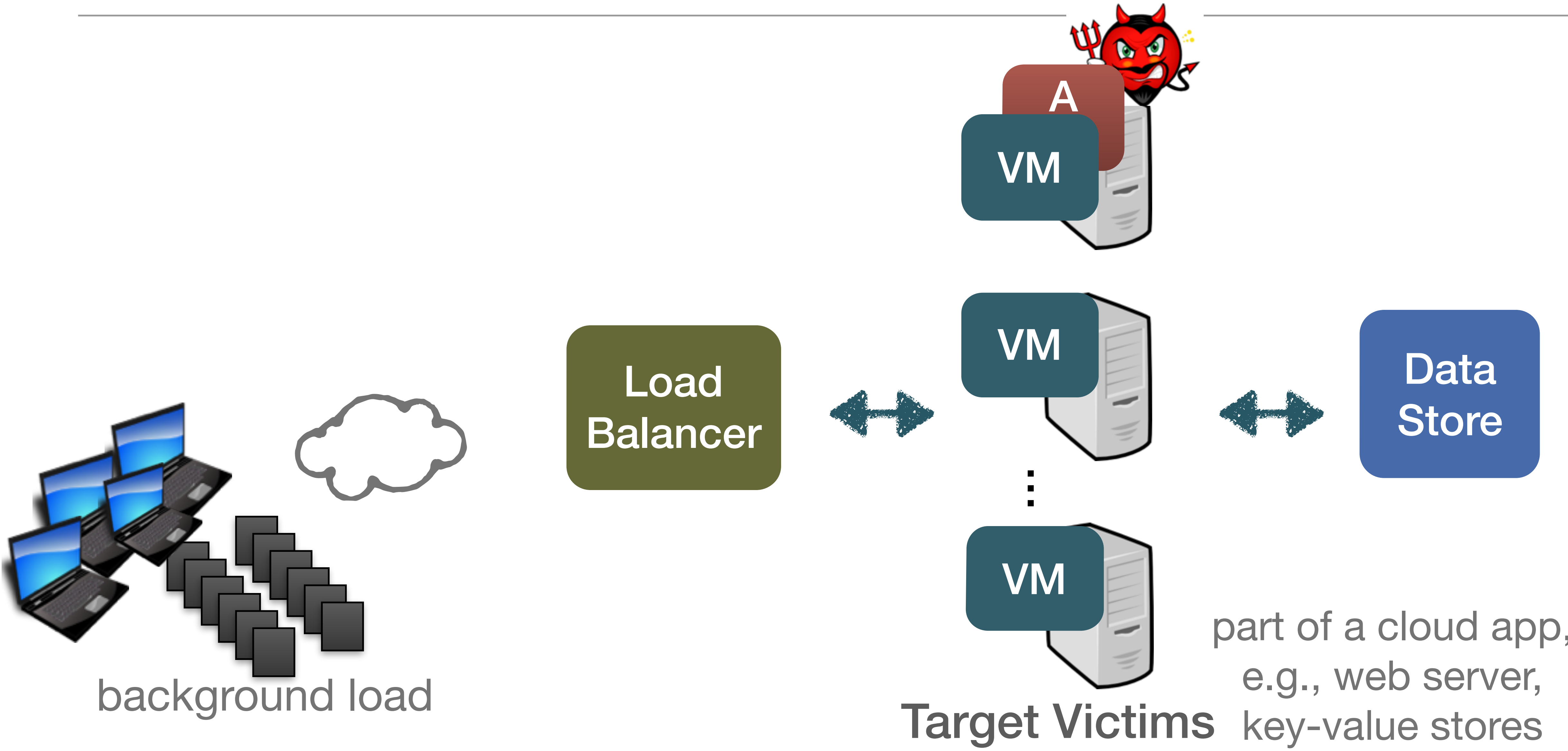
- Realistic victim setting: modern multi-tier cloud app.

Co-residency Detection on Uncooperative Victim



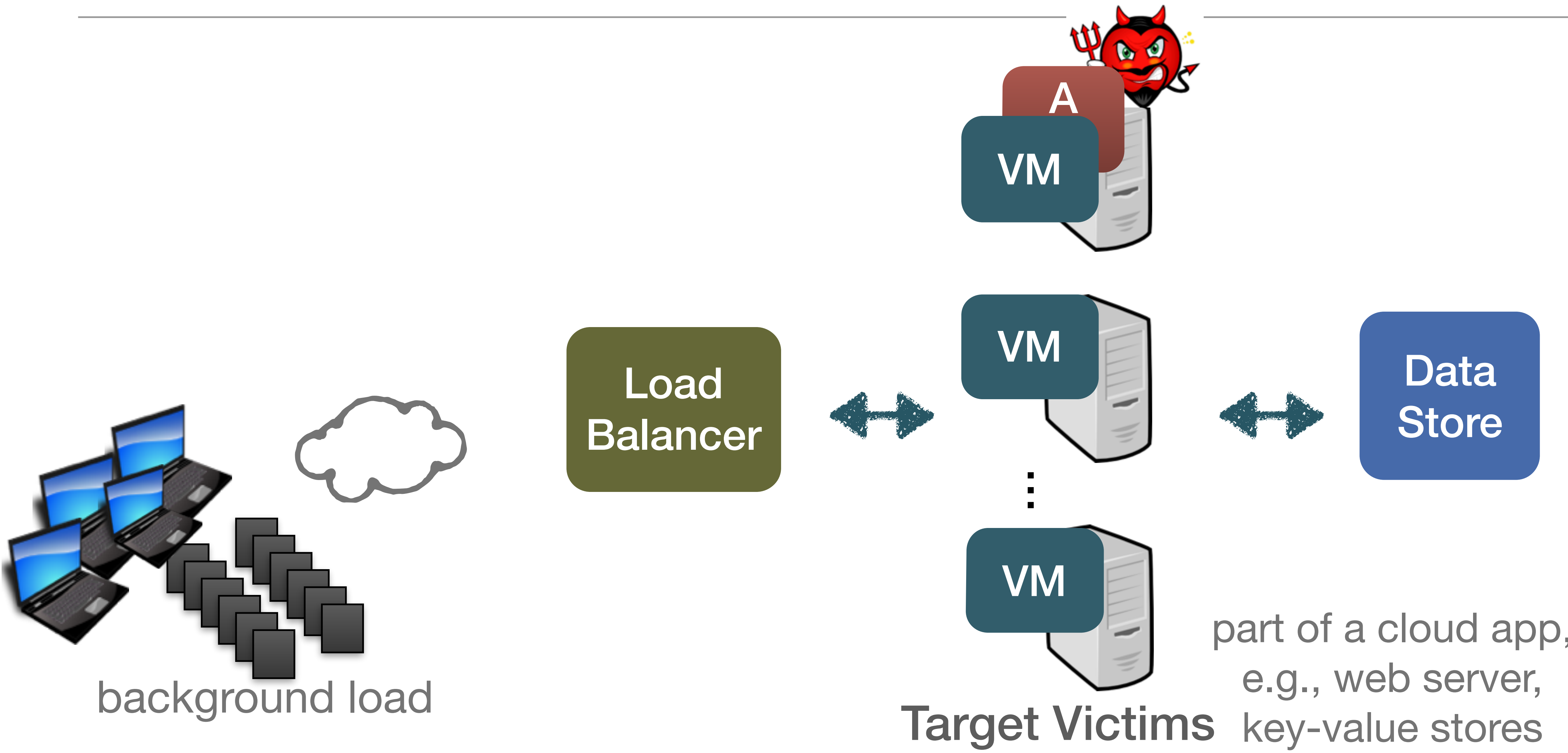
- Realistic victim setting: modern multi-tier cloud app.

Co-residency Detection on Uncooperative Victim



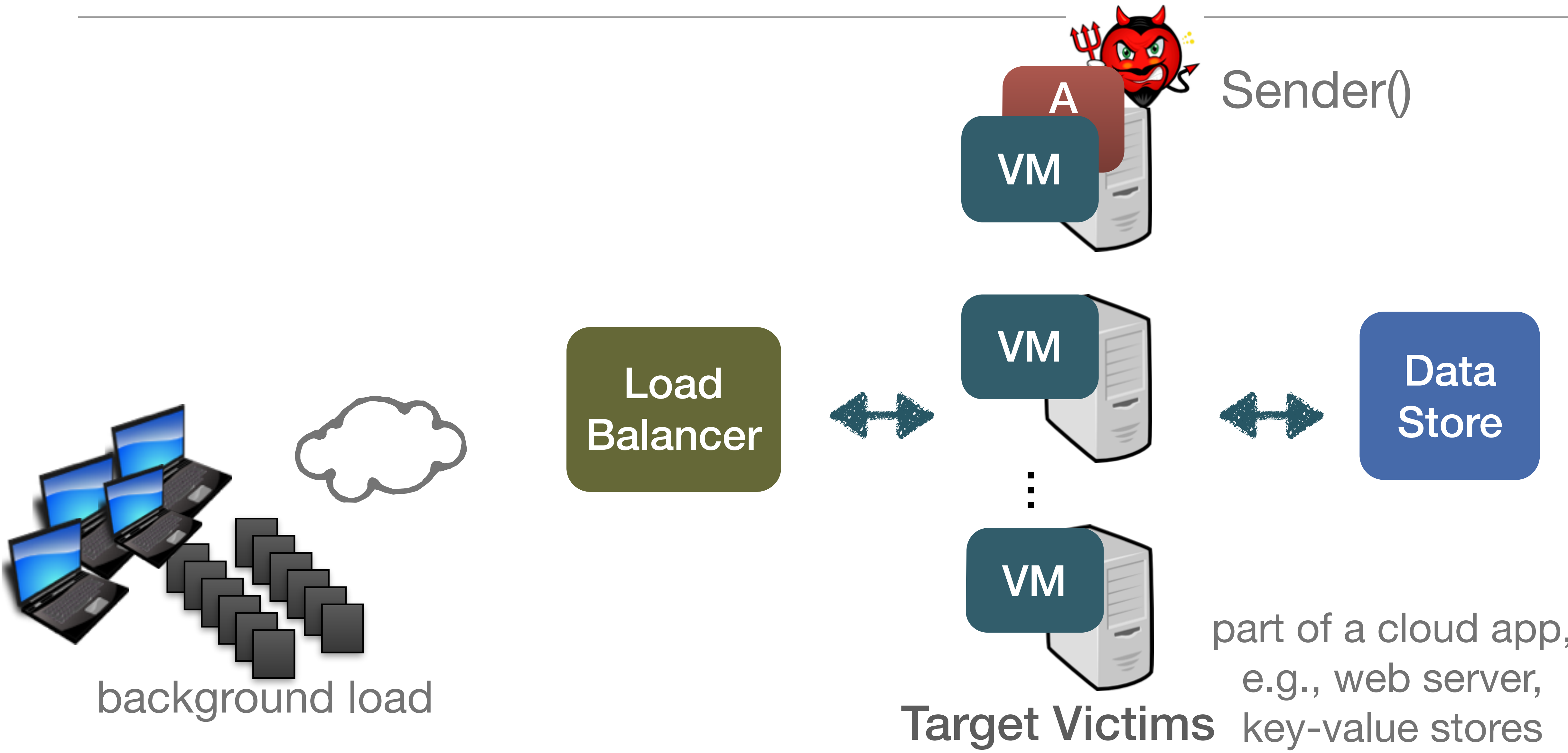
- Realistic victim setting: modern multi-tier cloud app.

Co-residency Detection on Uncooperative Victim



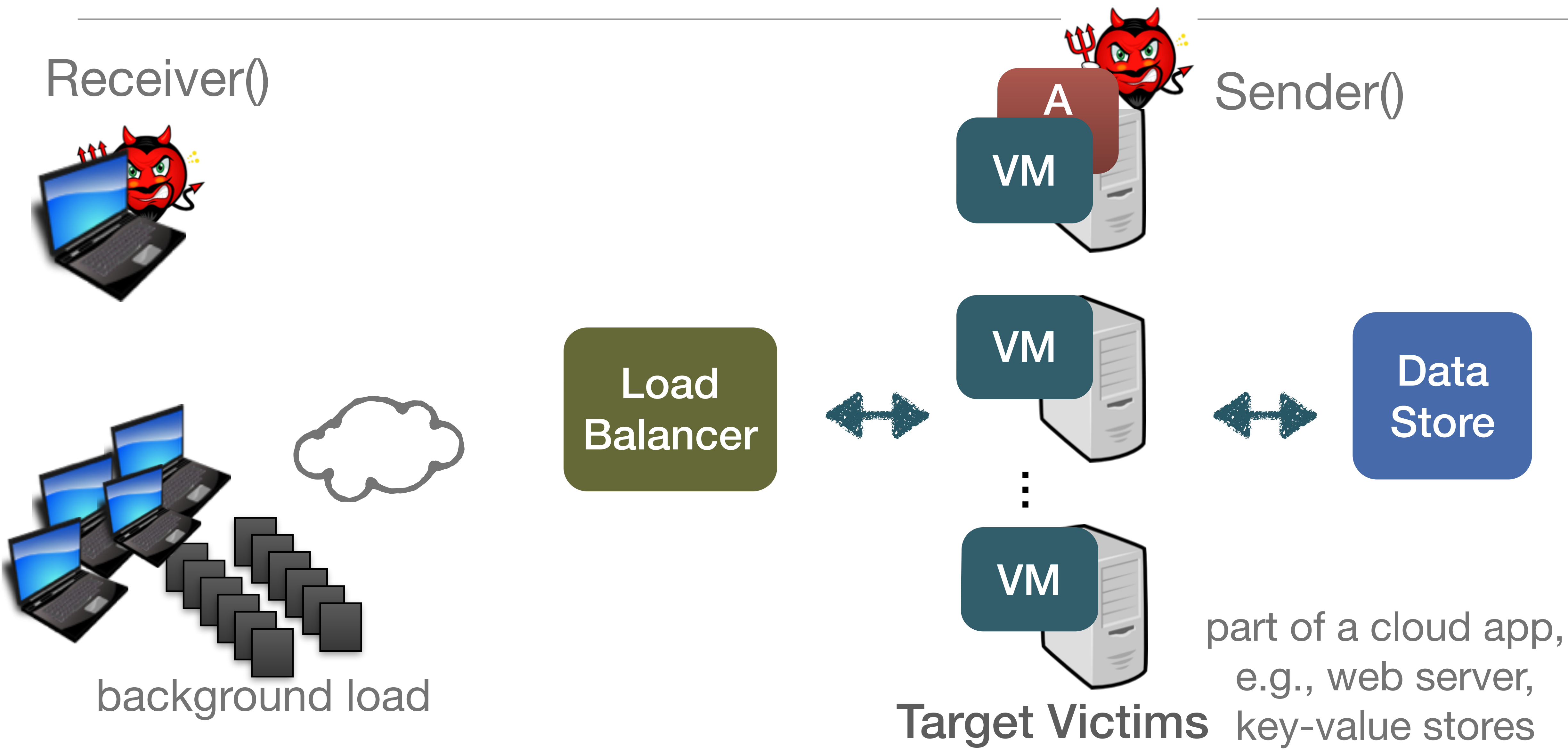
- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic

Co-residency Detection on Uncooperative Victim



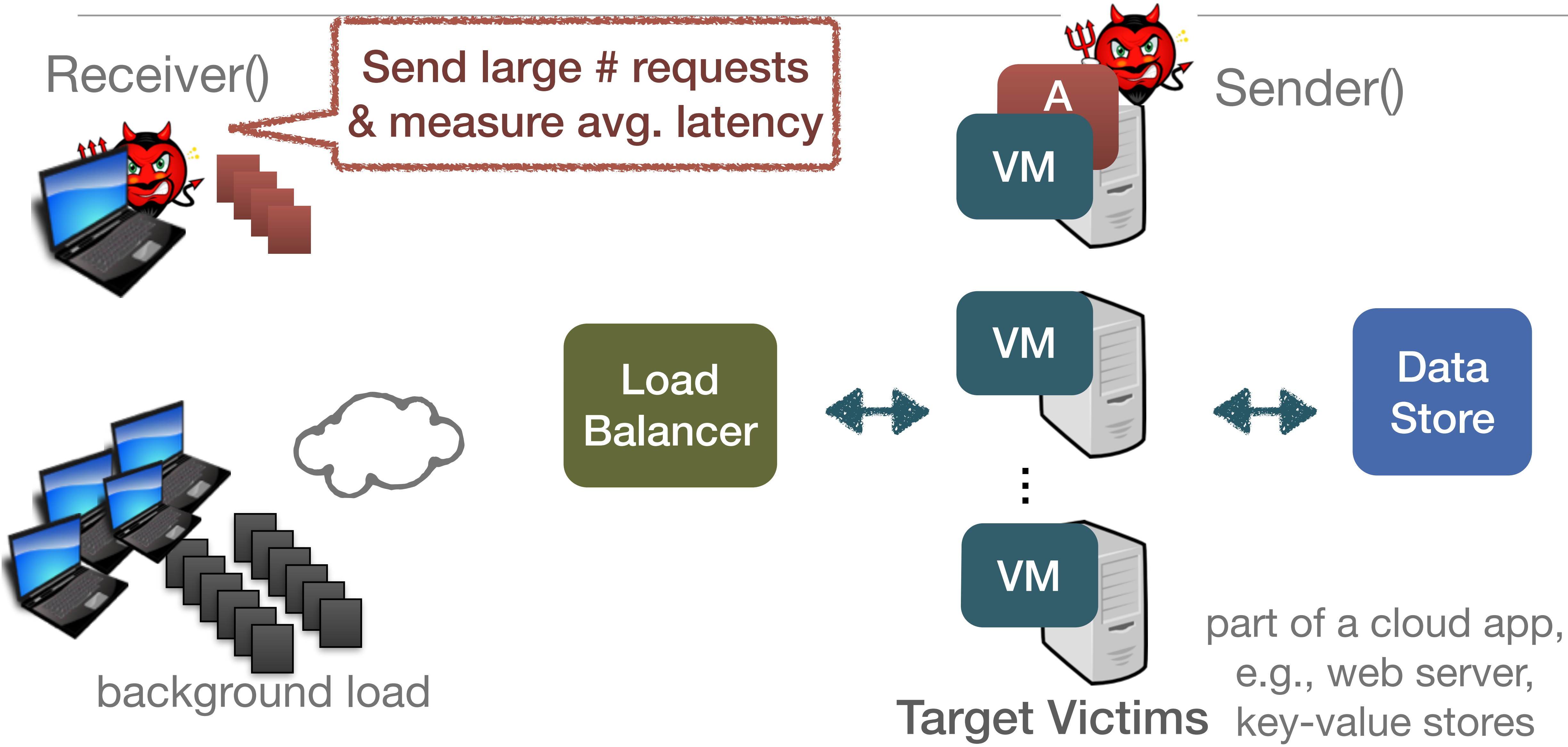
- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic

Co-residency Detection on Uncooperative Victim



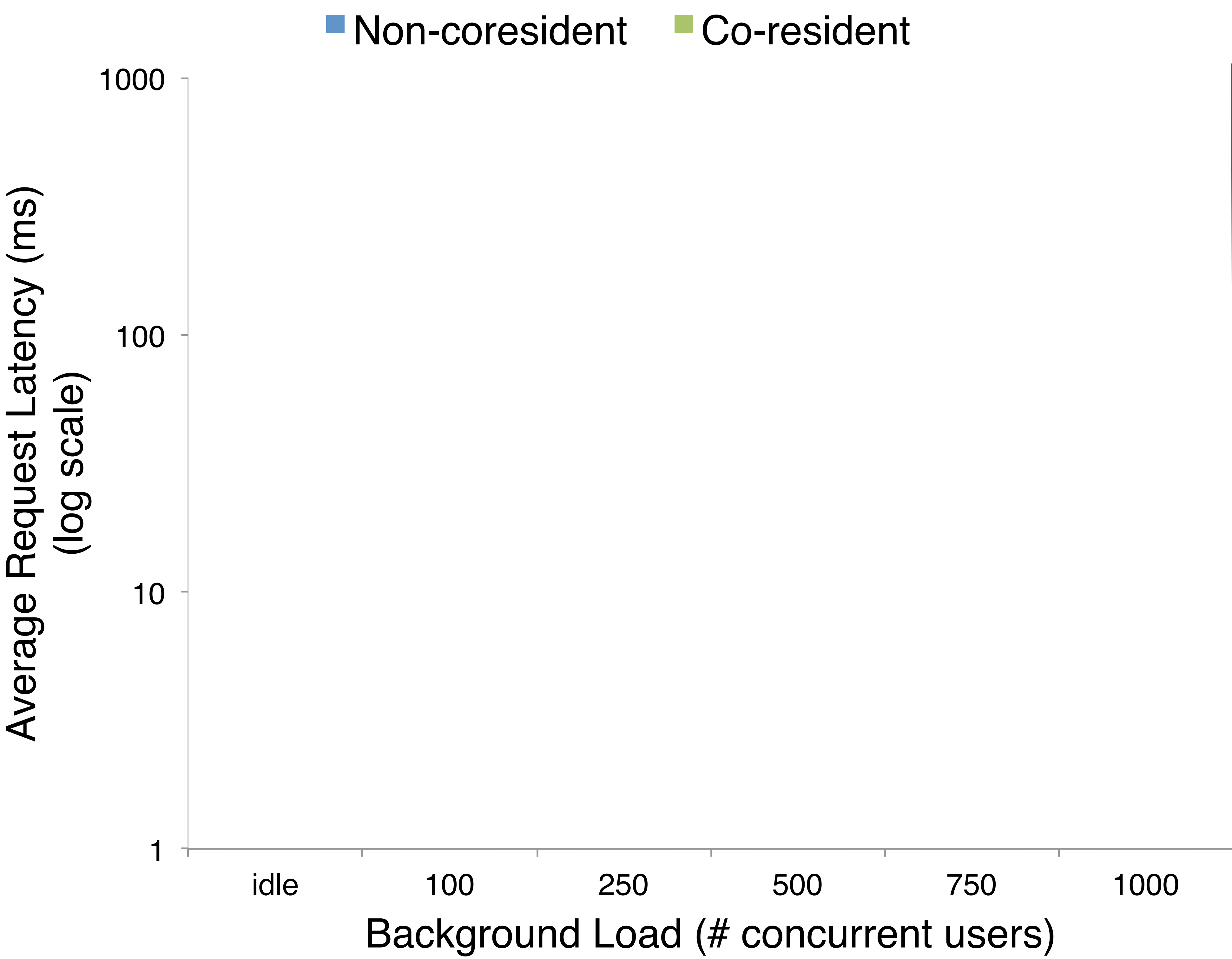
- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic

Co-residency Detection on Uncooperative Victim



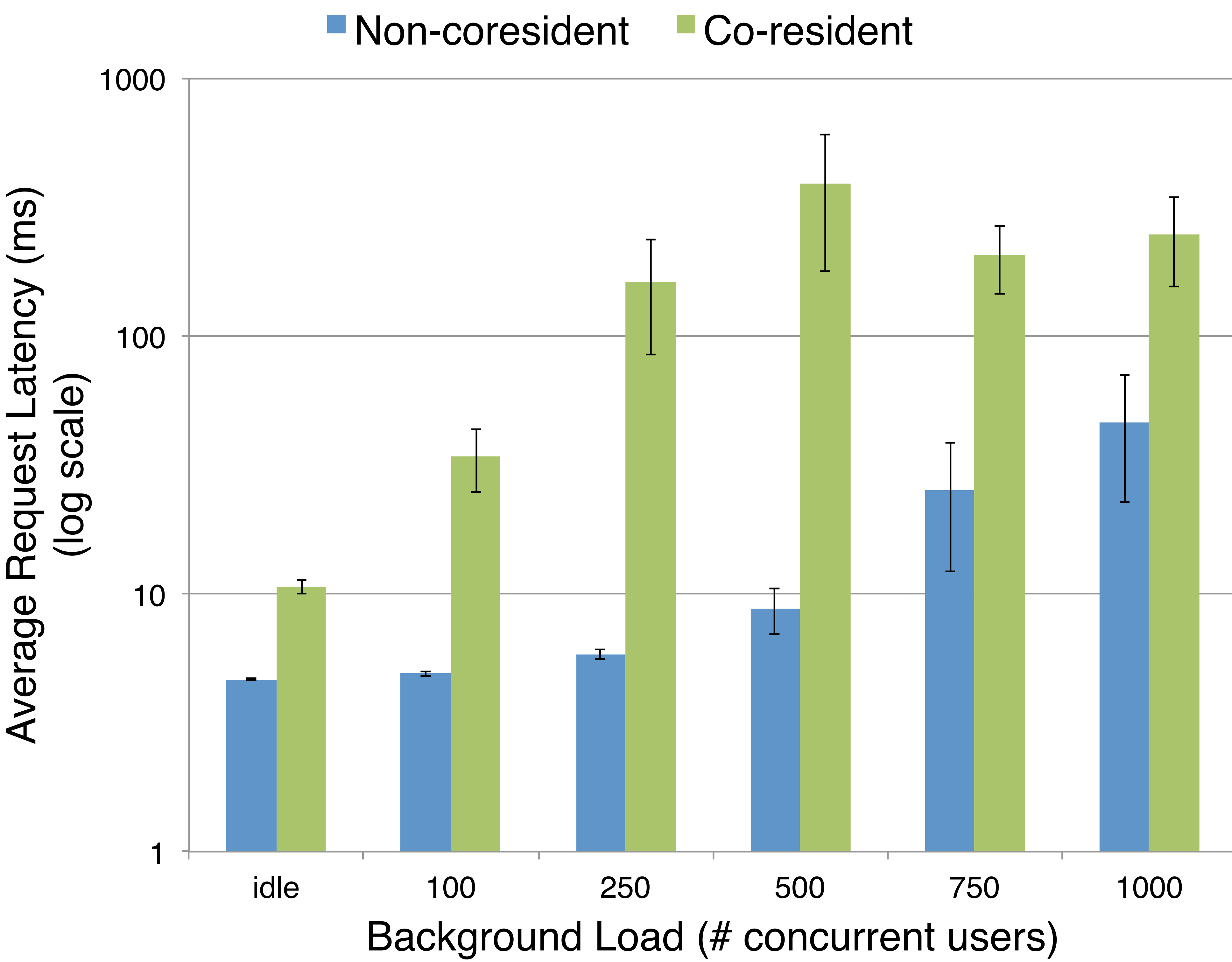
- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic

Co-residency Detection on Uncooperative Victim



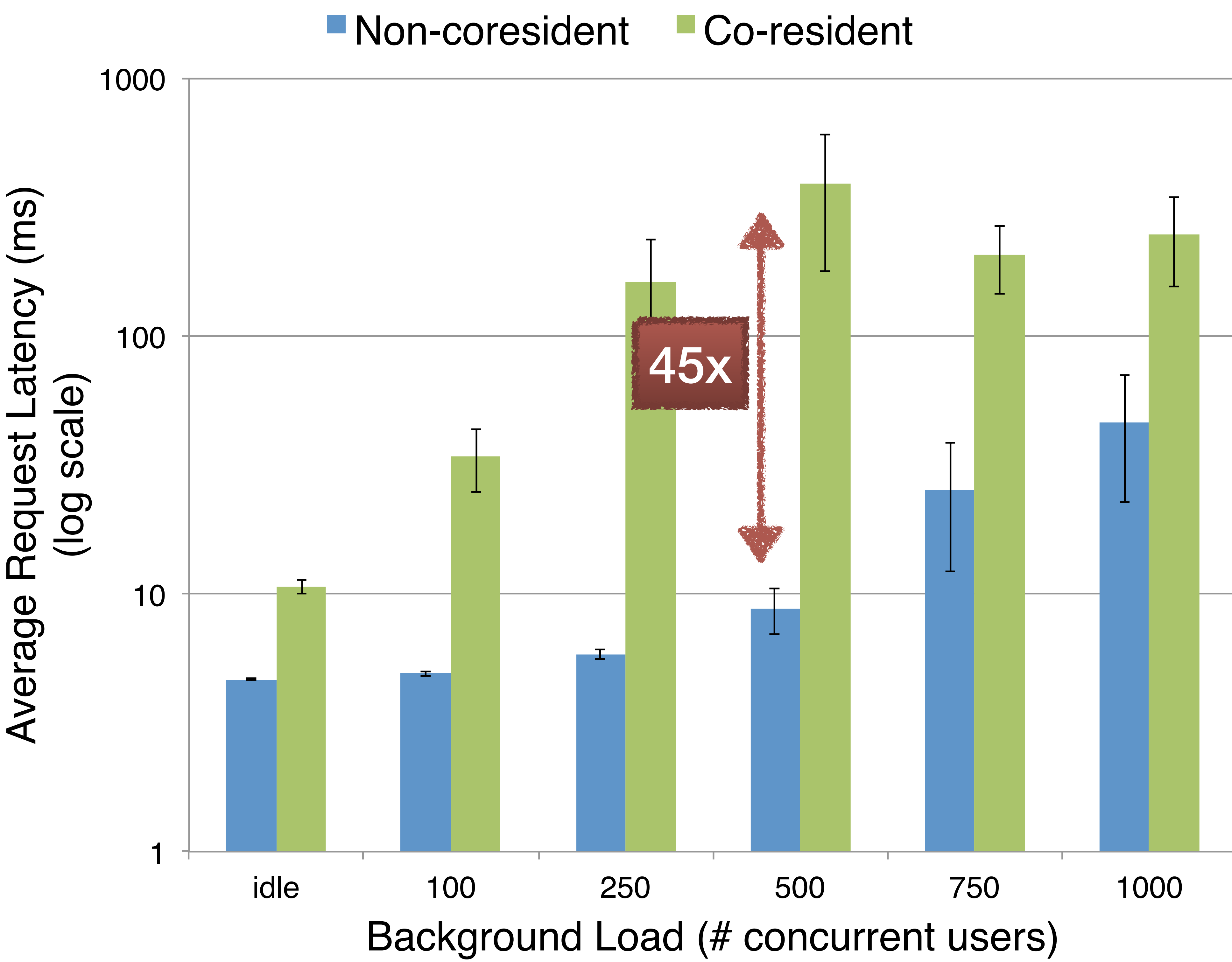
Social networking application (Olio):
HAProxy LB +
3 Web servers +
1 mysql server

Co-residency Detection on Uncooperative Victim



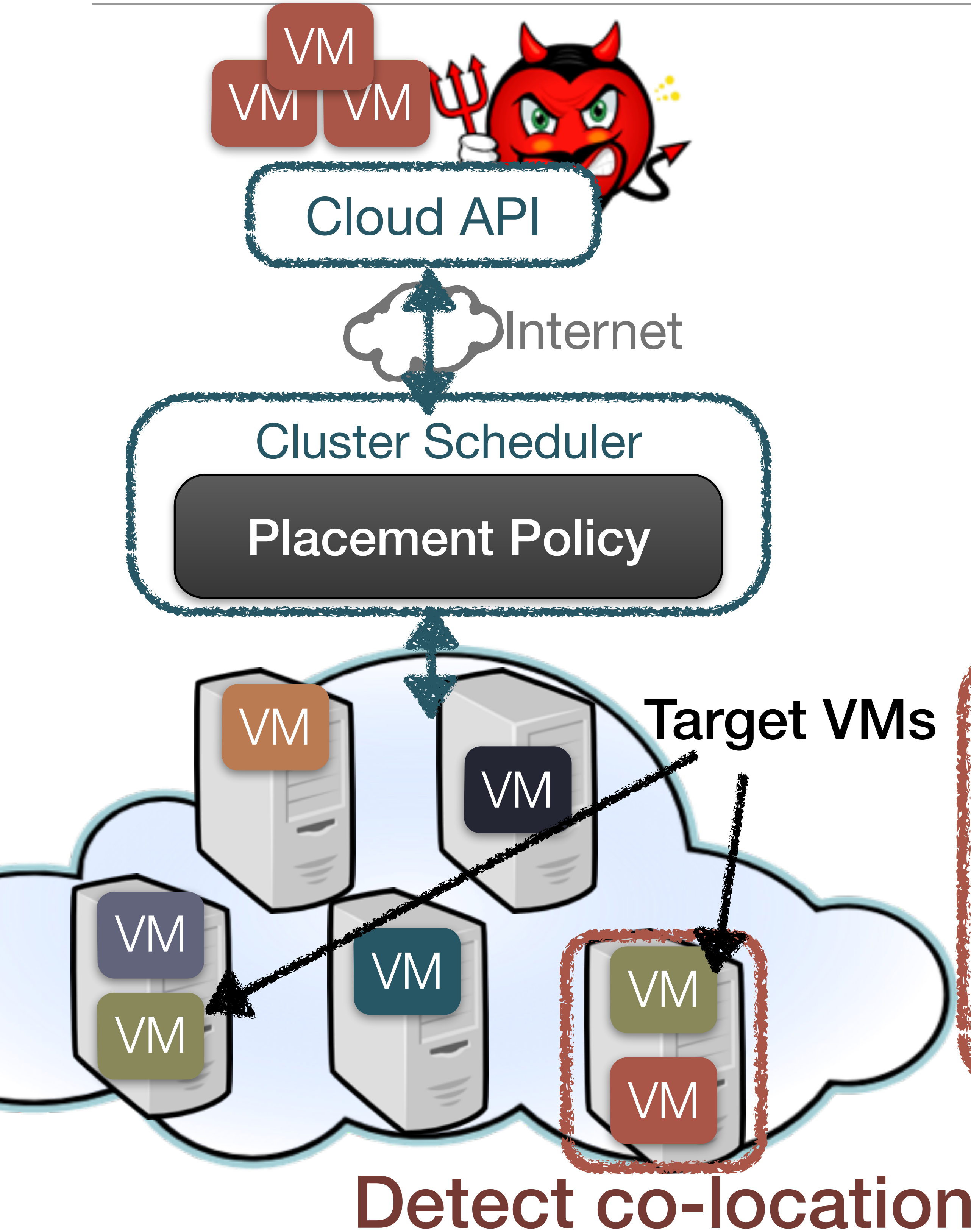
Social networking application (Olio):
HAProxy LB +
3 Web servers +
1 mysql server

Co-residency Detection on Uncooperative Victim



Social networking application (Olio): HAProxy LB + 3 Web servers + 1 mysql server

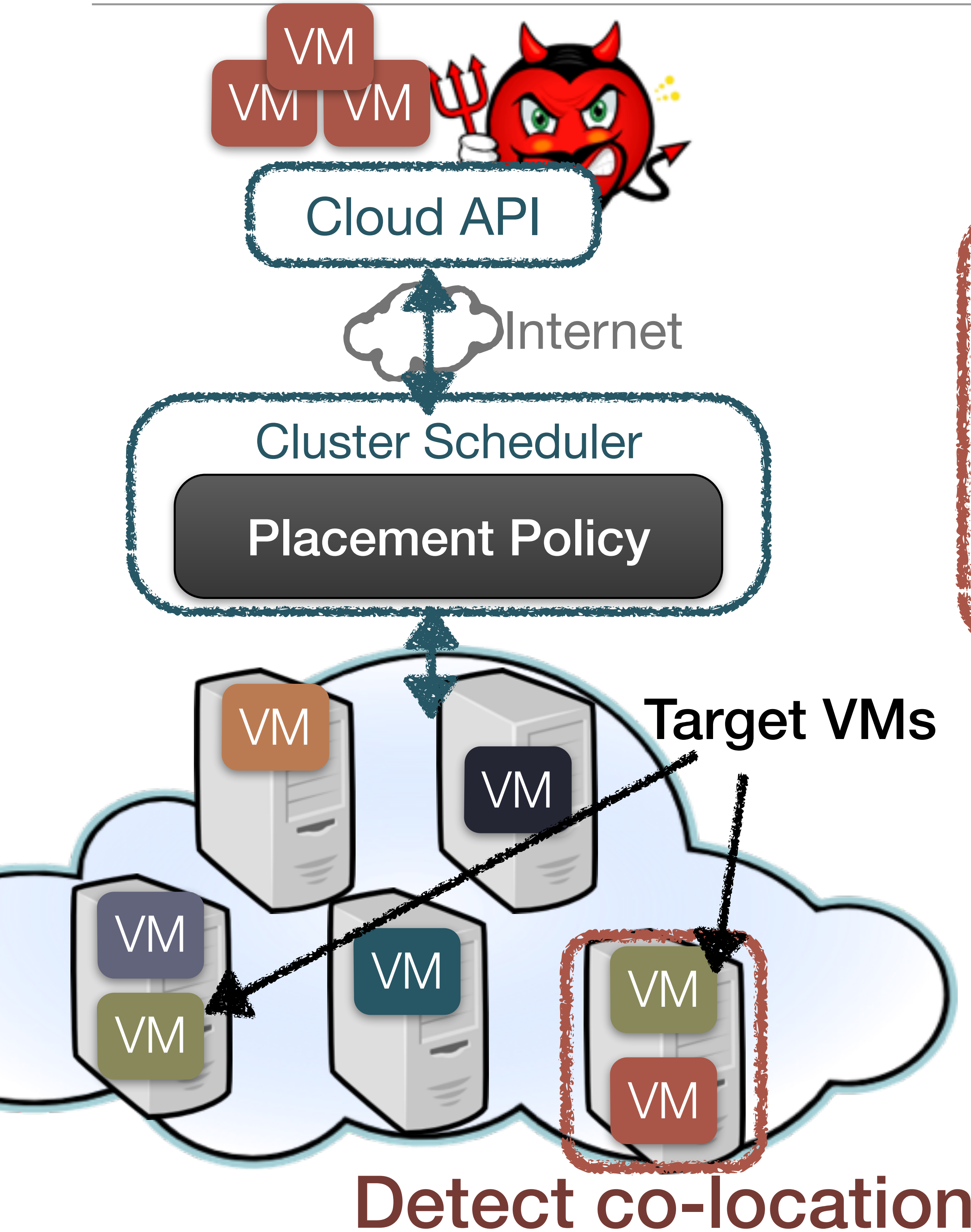
Our Work:
Exploring Co-location Attacks in Modern Clouds



Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location
2. Detecting Co-location
   - with any target victim

Our Work:
Exploring Co-location Attacks in Modern Clouds



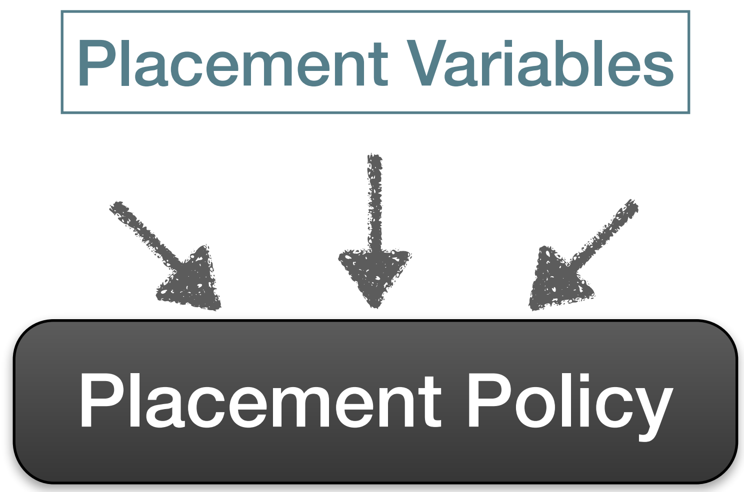
Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location
2. Detecting Co-location
   - with any target victim

Big Picture: Placement Vulnerability Study

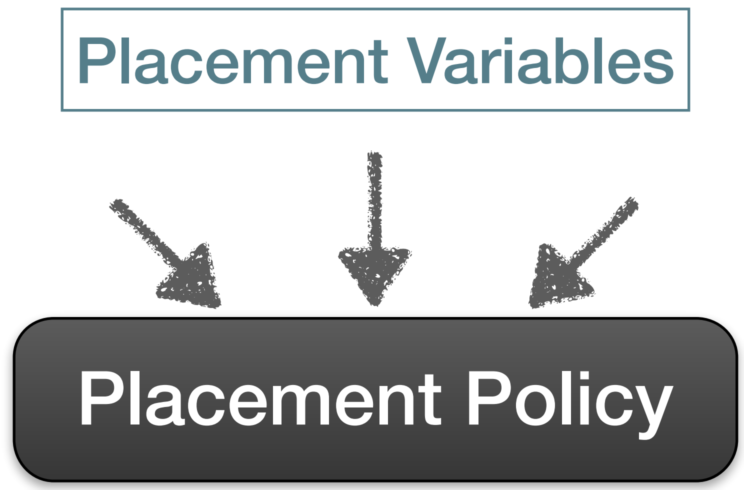
Placement Policy

Big Picture: Placement Vulnerability Study



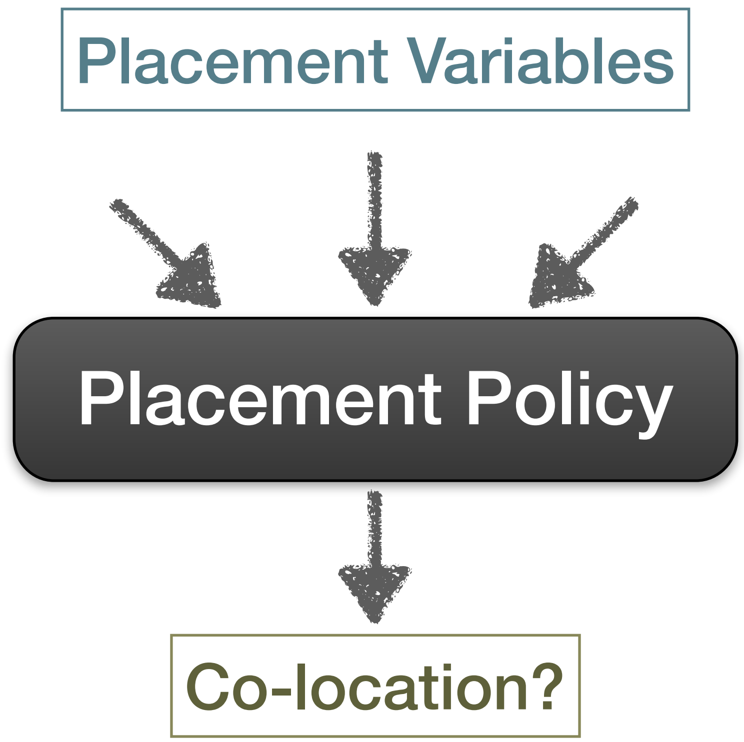
Big Picture: Placement Vulnerability Study

e.g., # VMs, when you launch, datacenter, VM type, etc.



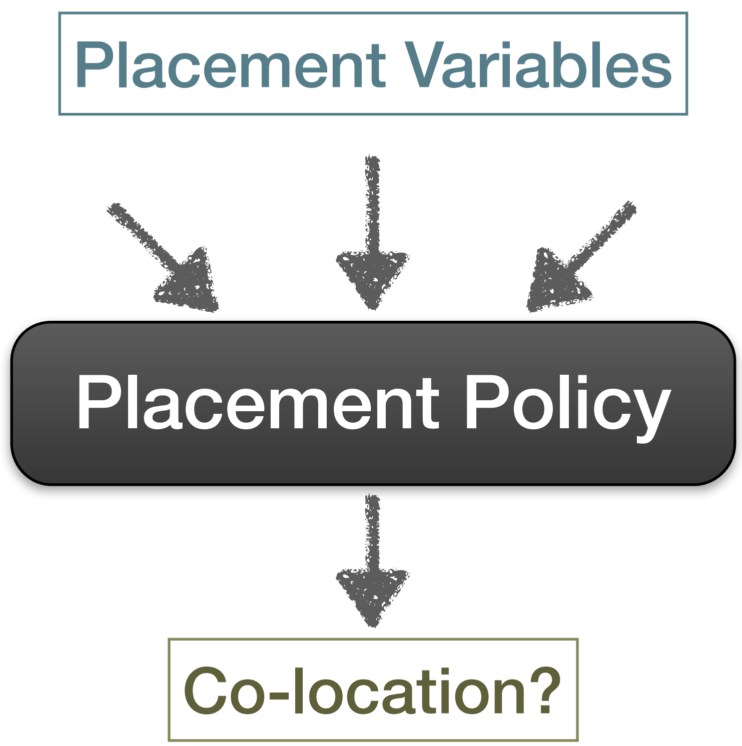
Big Picture: Placement Vulnerability Study

e.g., # VMs, when you launch, datacenter, VM type, etc.



Big Picture: Placement Vulnerability Study

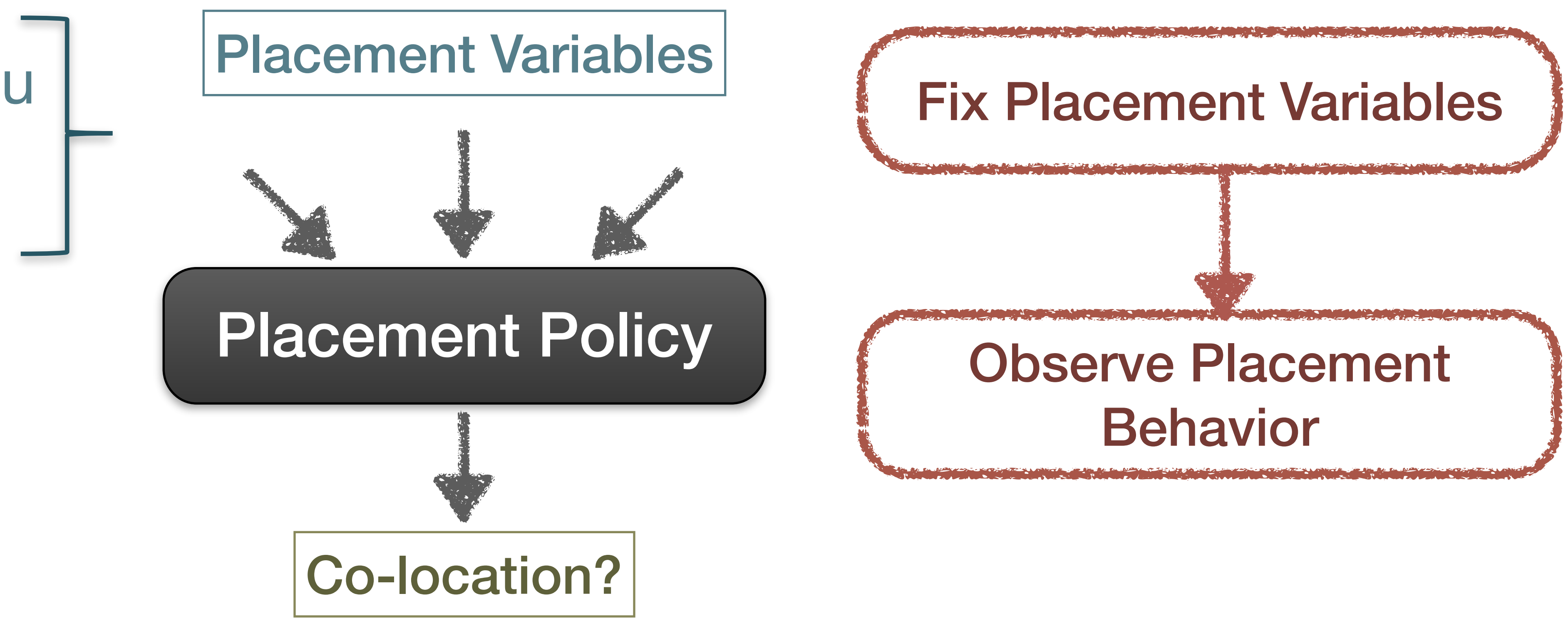
e.g., # VMs, when you launch, datacenter, VM type, etc.



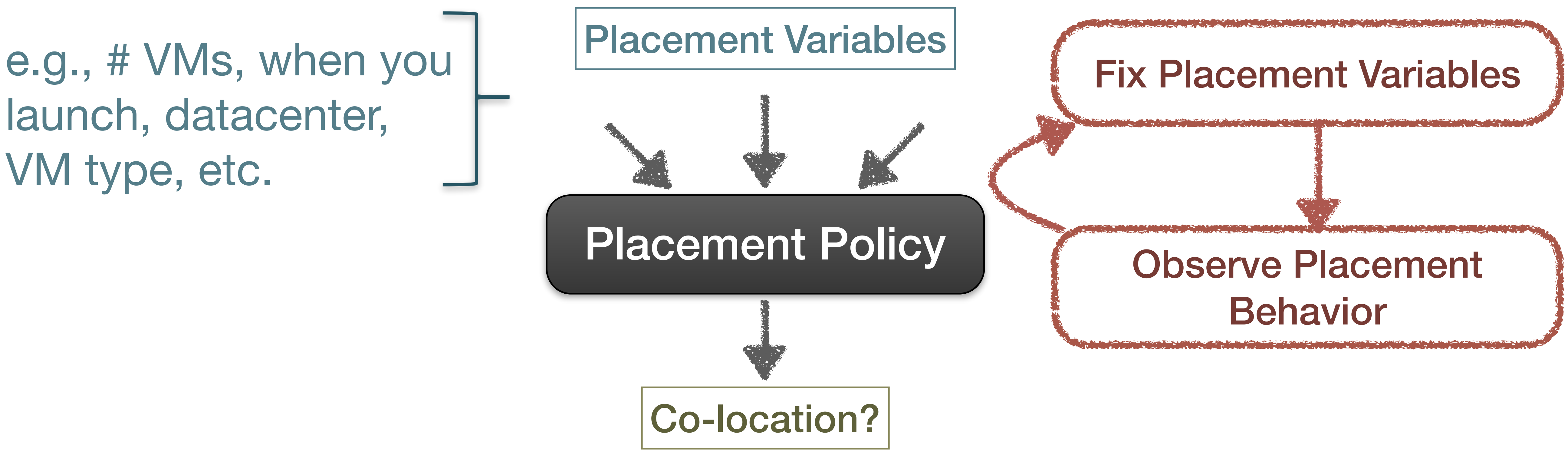
Fix Placement Variables

Big Picture: Placement Vulnerability Study

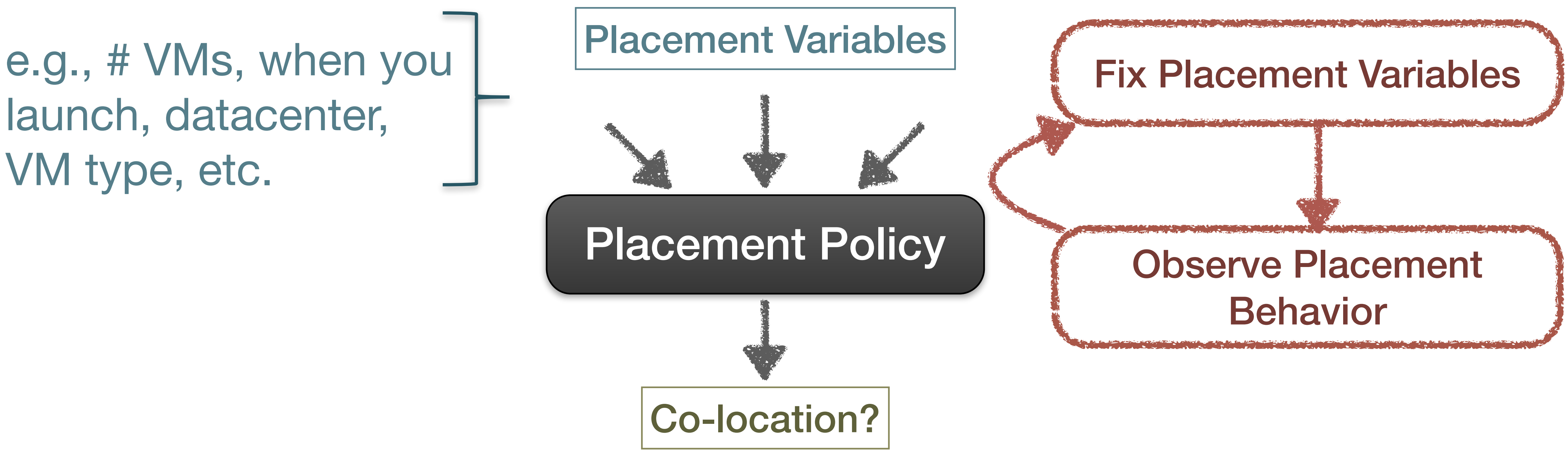
e.g., # VMs, when you launch, datacenter, VM type, etc.



Big Picture: Placement Vulnerability Study

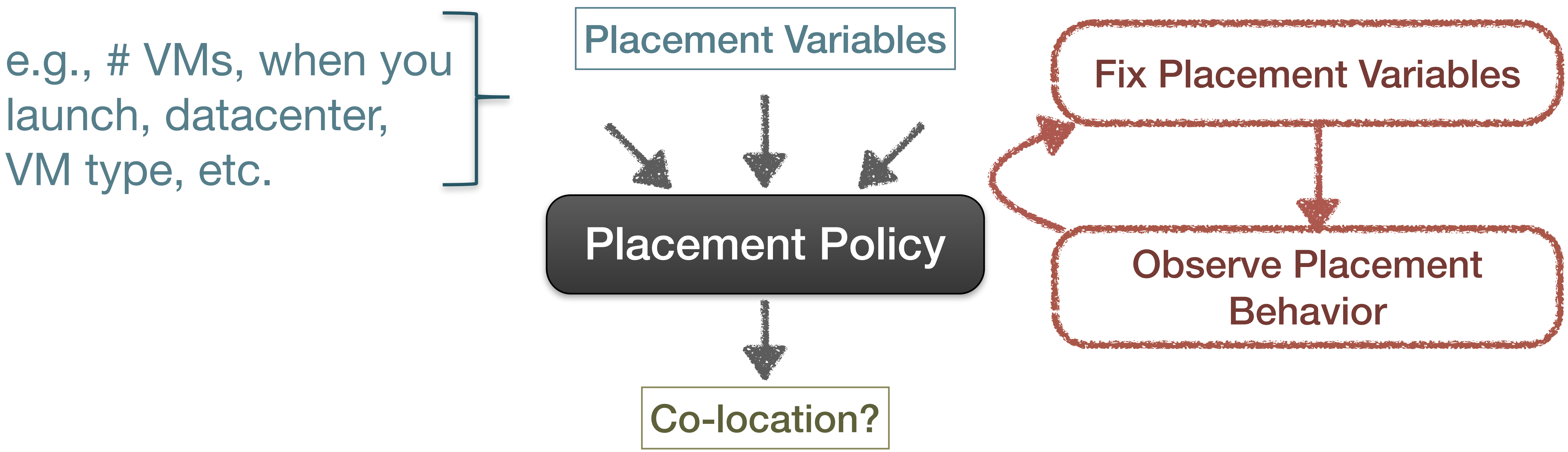


Big Picture: Placement Vulnerability Study



Study spanning 3 months, exploring 6 placement variables, spending more than $200 per cloud

Big Picture: Placement Vulnerability Study



Study spanning 3 months, exploring 6 placement variables, spending more than $200 per cloud

Google Compute Engine

amazon web services™ EC2

Microsoft Azure

Study Setup

- Two distinct accounts: proxy for victim and attacker

Google Compute Engine

amazon web services™ EC2

Microsoft Azure

Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
  - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
  - Small instance type
    (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
  - Values for these variables form
    a launch strategy

Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
  - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
  - Small instance type (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
  - Values for these variables form a launch strategy
- Execute a launch strategy from a workstation
  - detect and log co-location

Google Compute Engine

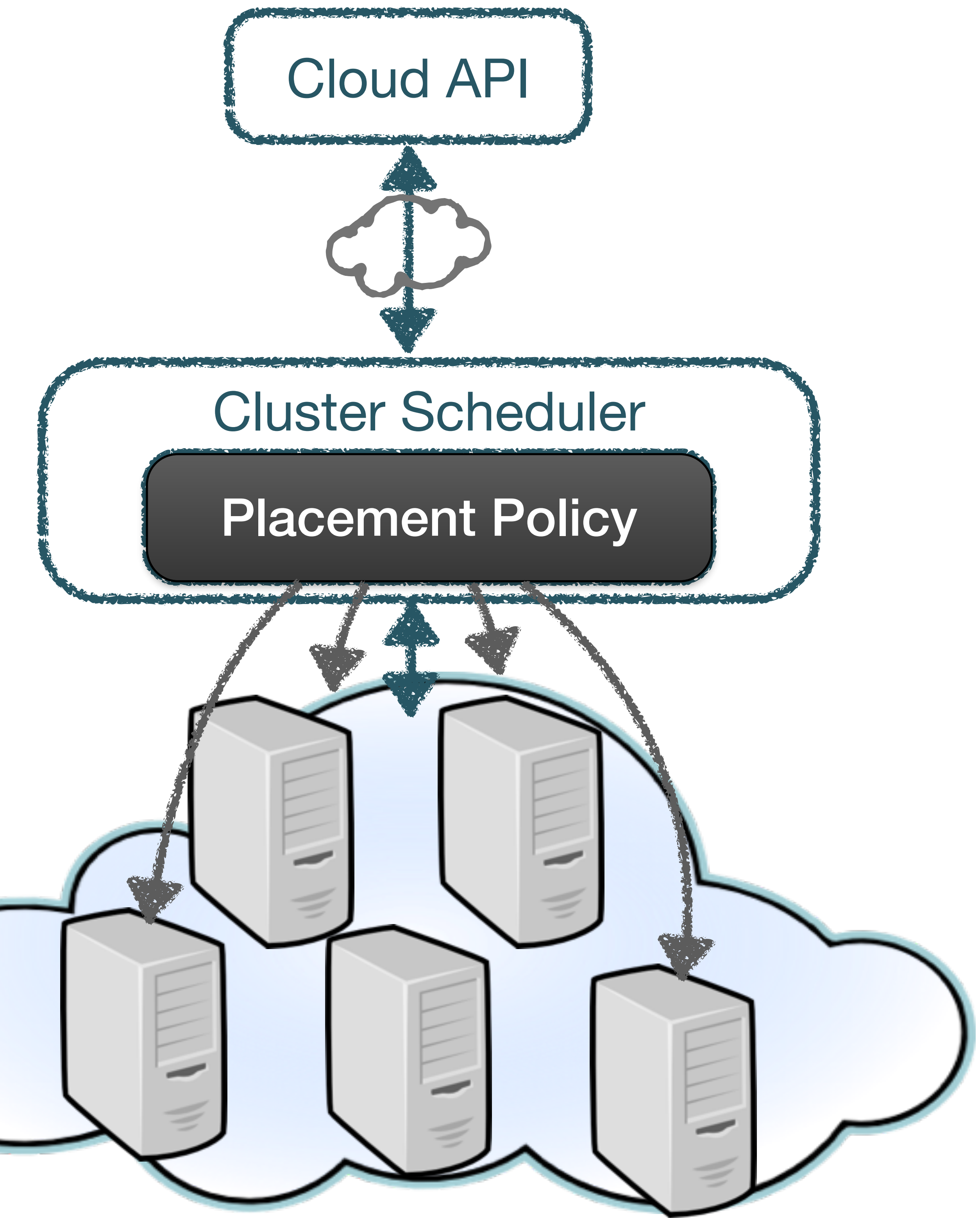
amazon web services™ EC2

Microsoft Azure

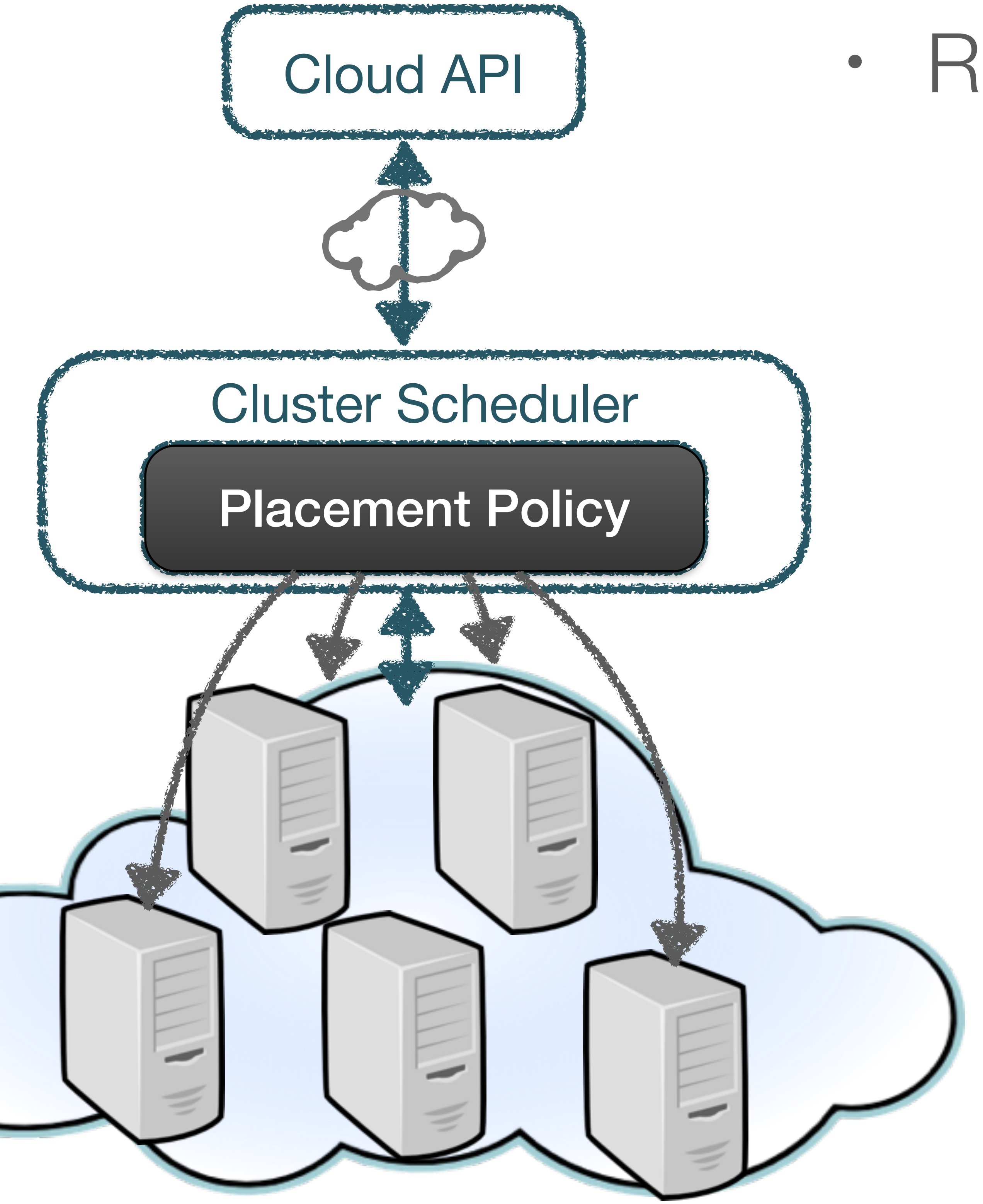
Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
    - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
    - Small instance type
      (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
    - Values for these variables form
      a launch strategy
- Execute a launch strategy from a workstation
    - detect and log co-location
- 9 samples per strategy with 3 runs per time of day & 2 days of week (weekday/weekend)

How hard should it be to achieve co-location?

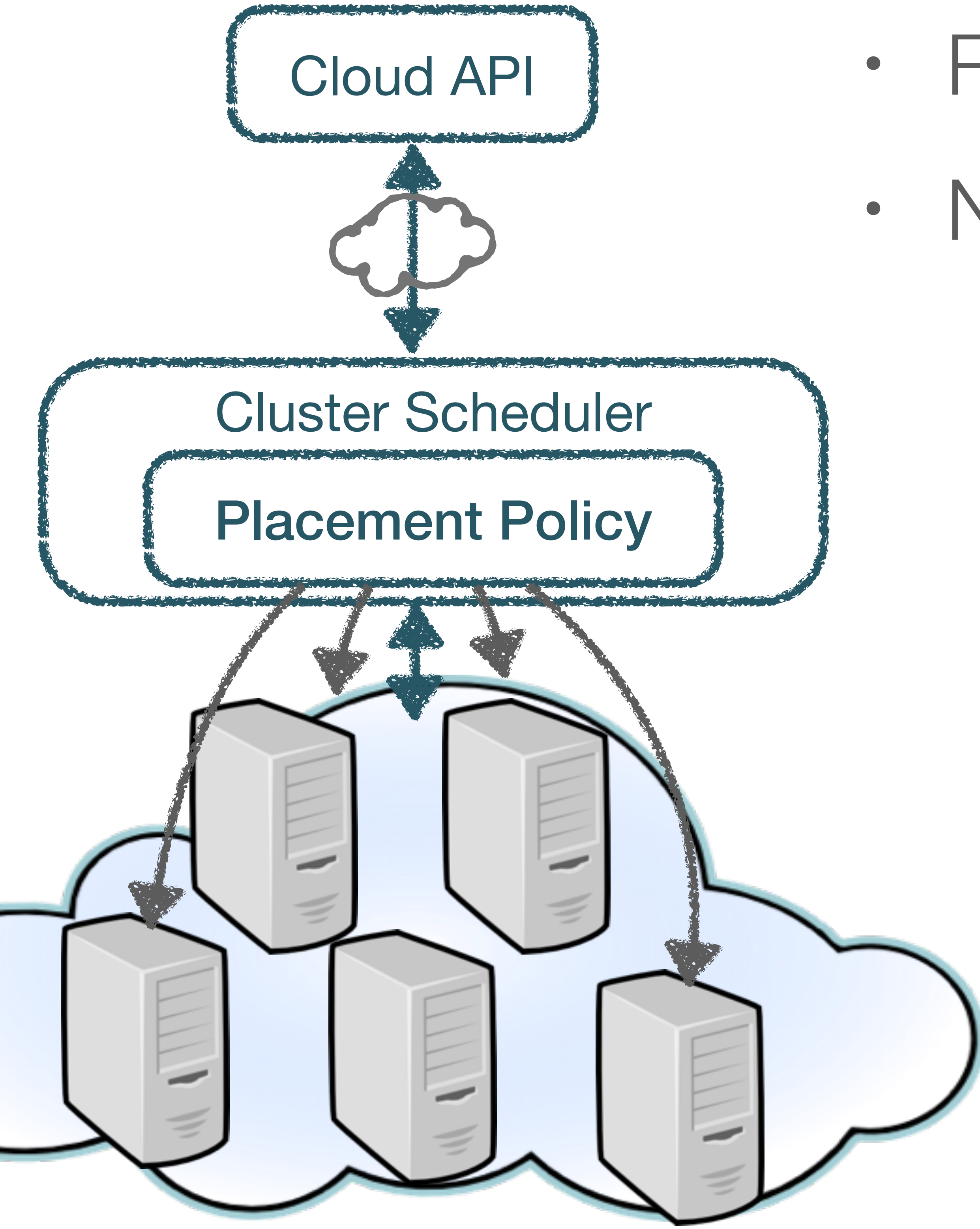


How hard should it be to achieve co-location?



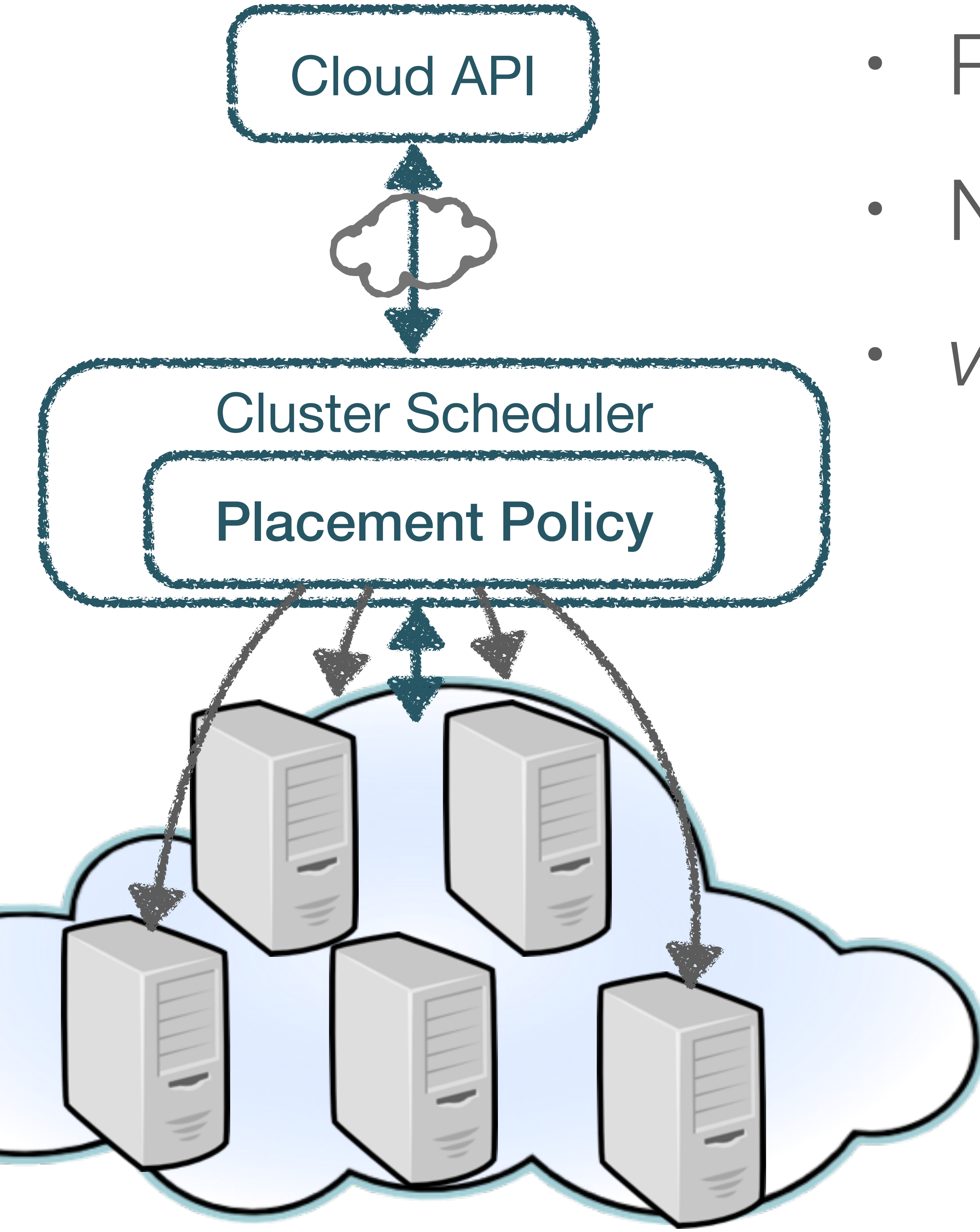
- Random placement policy

How hard should it be to achieve co-location?



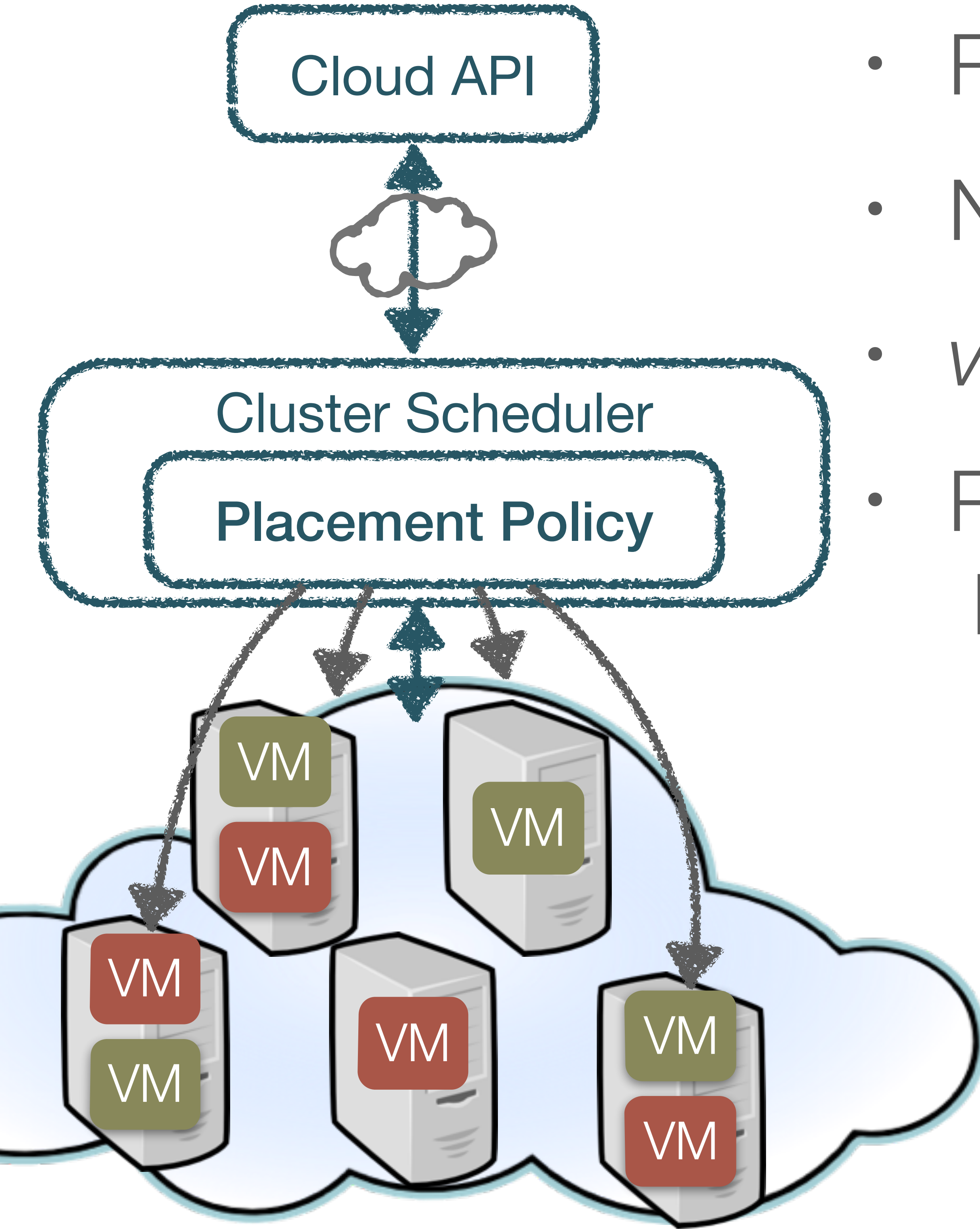
- Random placement policy
- N = 50,000 machines [re:Invent'14]

How hard should it be to achieve co-location?



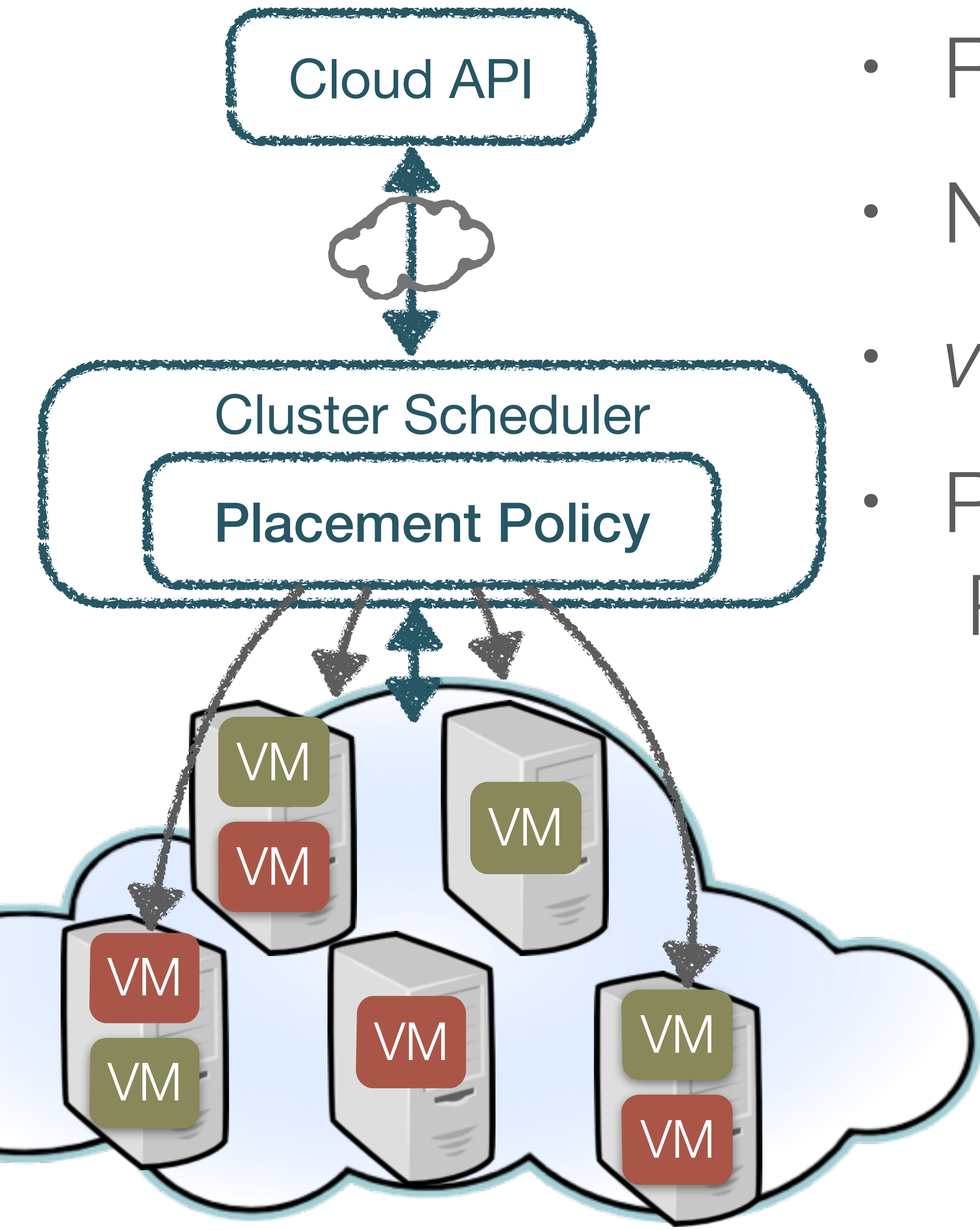
- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs

How hard should it be to achieve co-location?



- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs
- Probability of Collision:
  $P_c = 1 - (1 - v/N)^a$

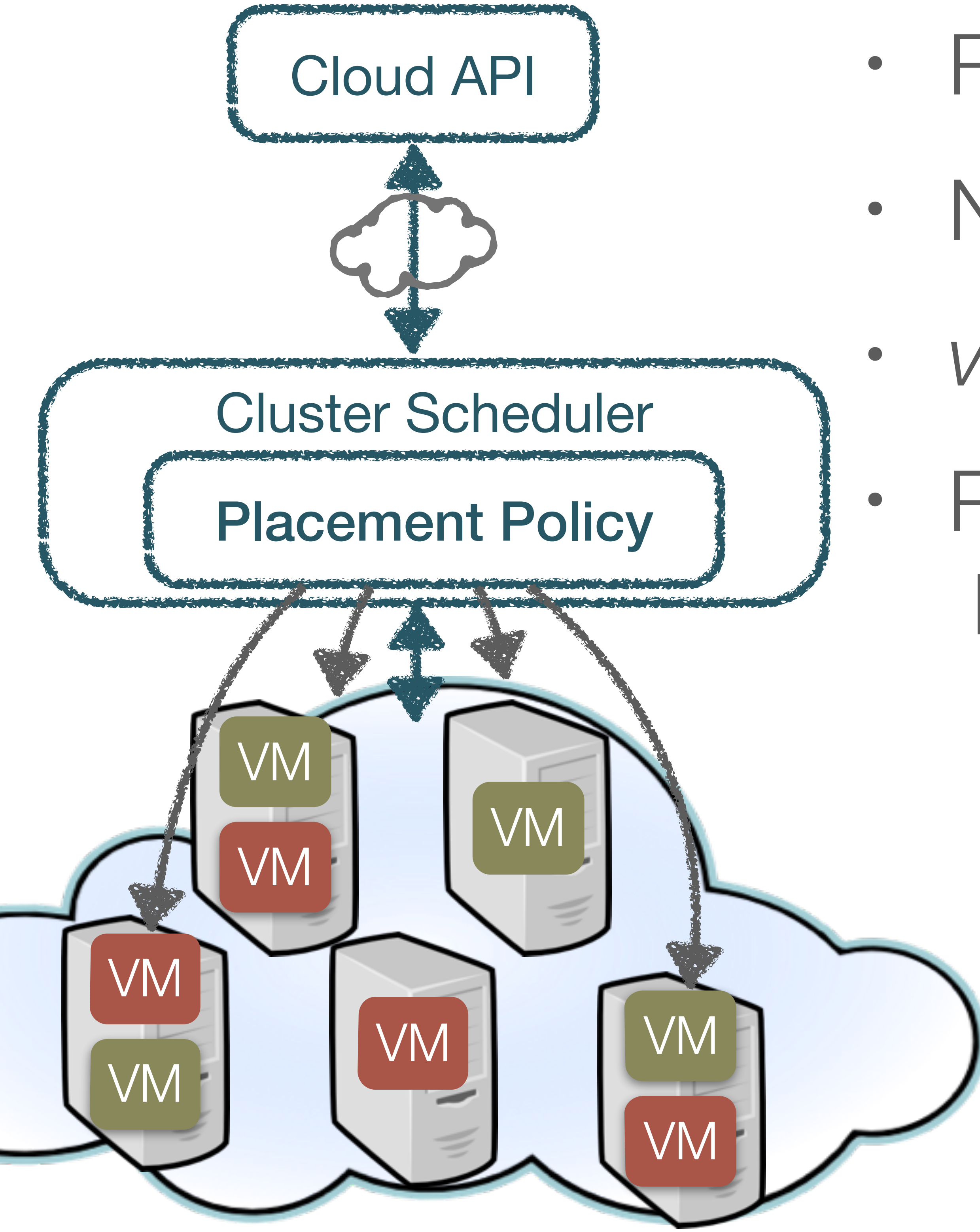
How hard should it be to achieve co-location?



- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs
- Probability of Collision:
  $P_c = 1 - (1 - v/N)^a$

v	$a = \ln(1-P_c)/\ln(1-v/N)$; $P_c = 0.5$

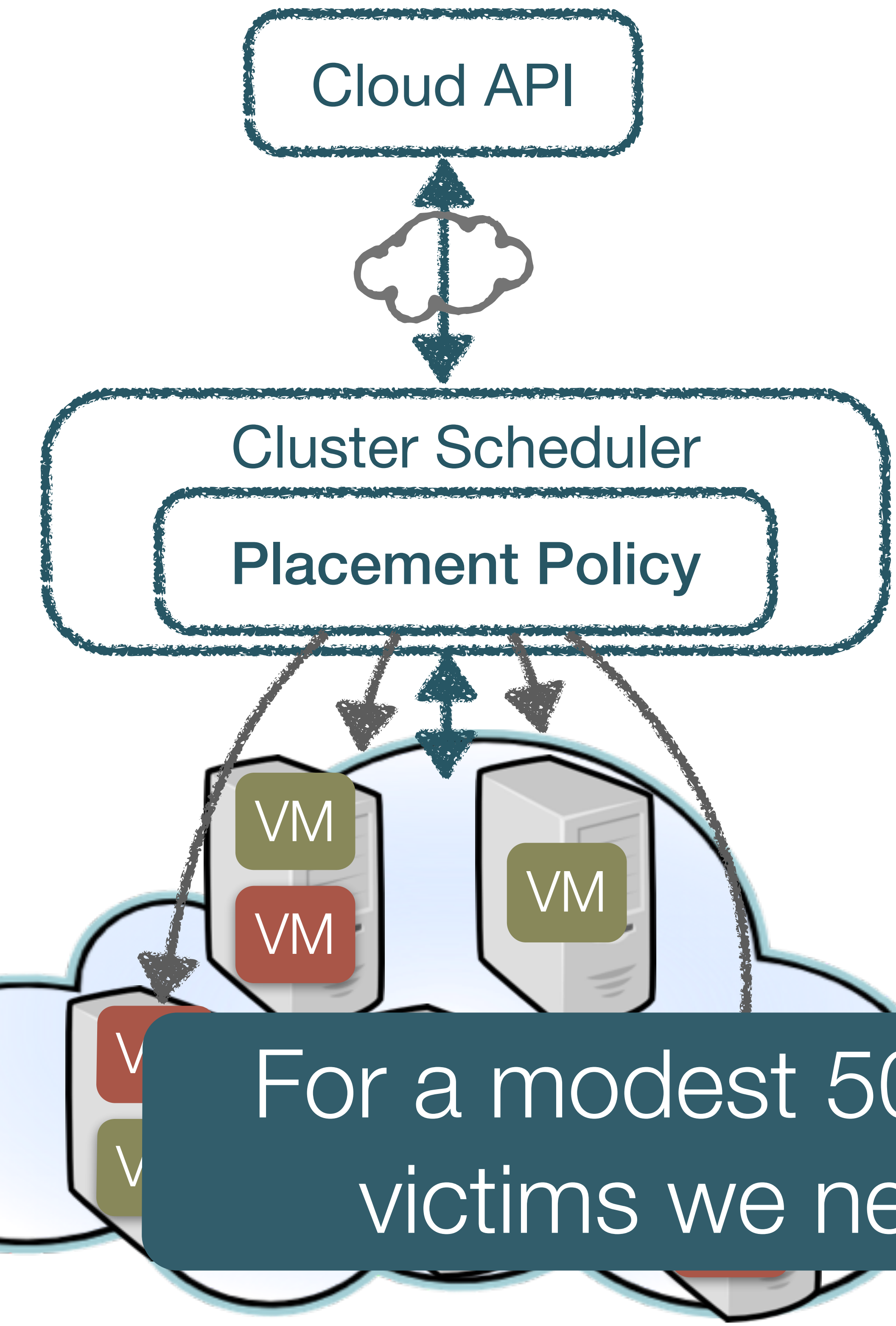
How hard should it be to achieve co-location?



- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs
- Probability of Collision:
  $P_c = 1 - (1 - v/N)^a$

v	$a = \ln(1 - P_c)/\ln(1 - v/N)$; $P_c = 0.5$
10	**3466**
20	**1733**
30	**1155**

How hard should it be to achieve co-location?

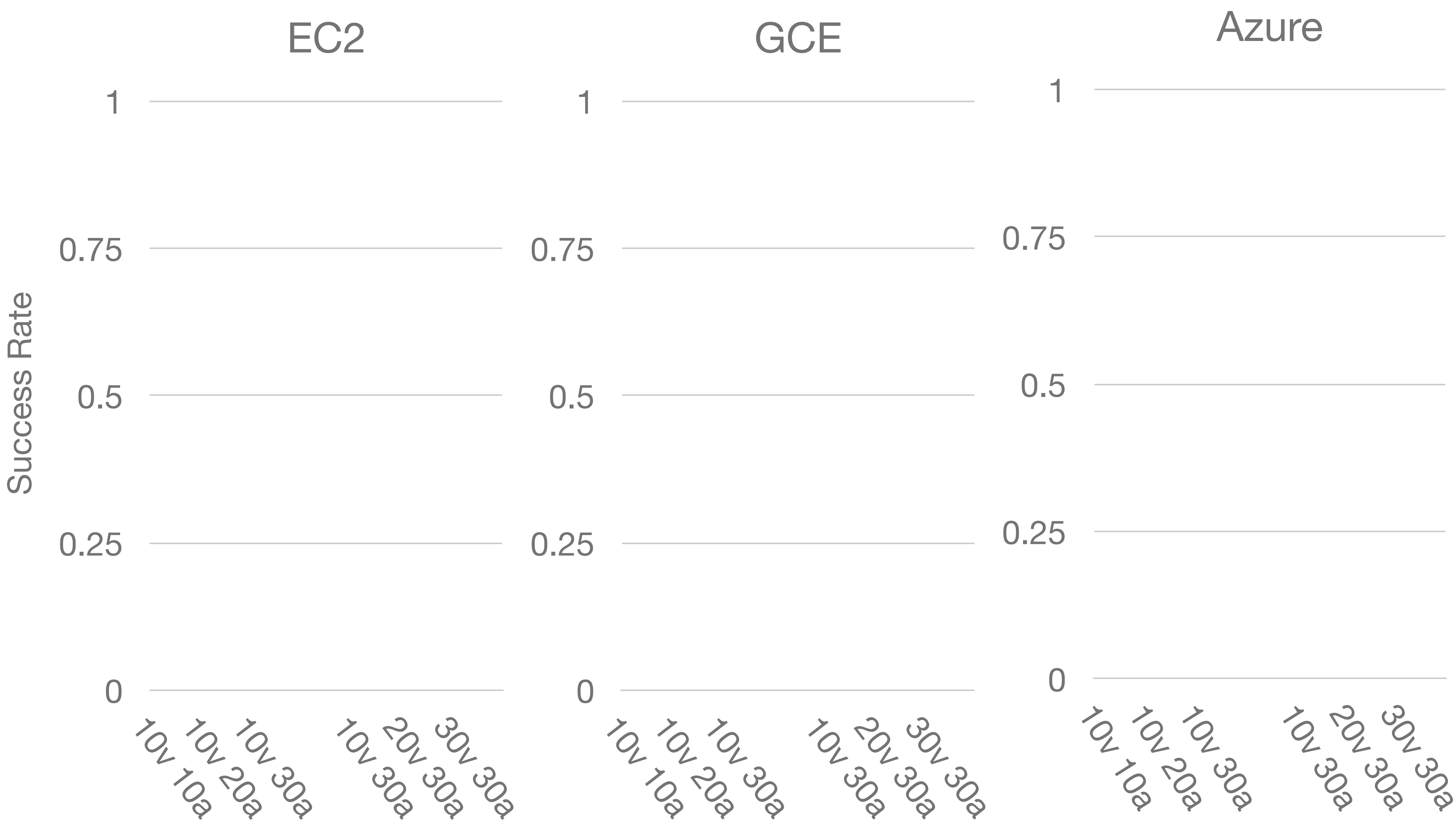


- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs
- Probability of Collision:
  $P_C = 1 - (1 - v/N)^a$

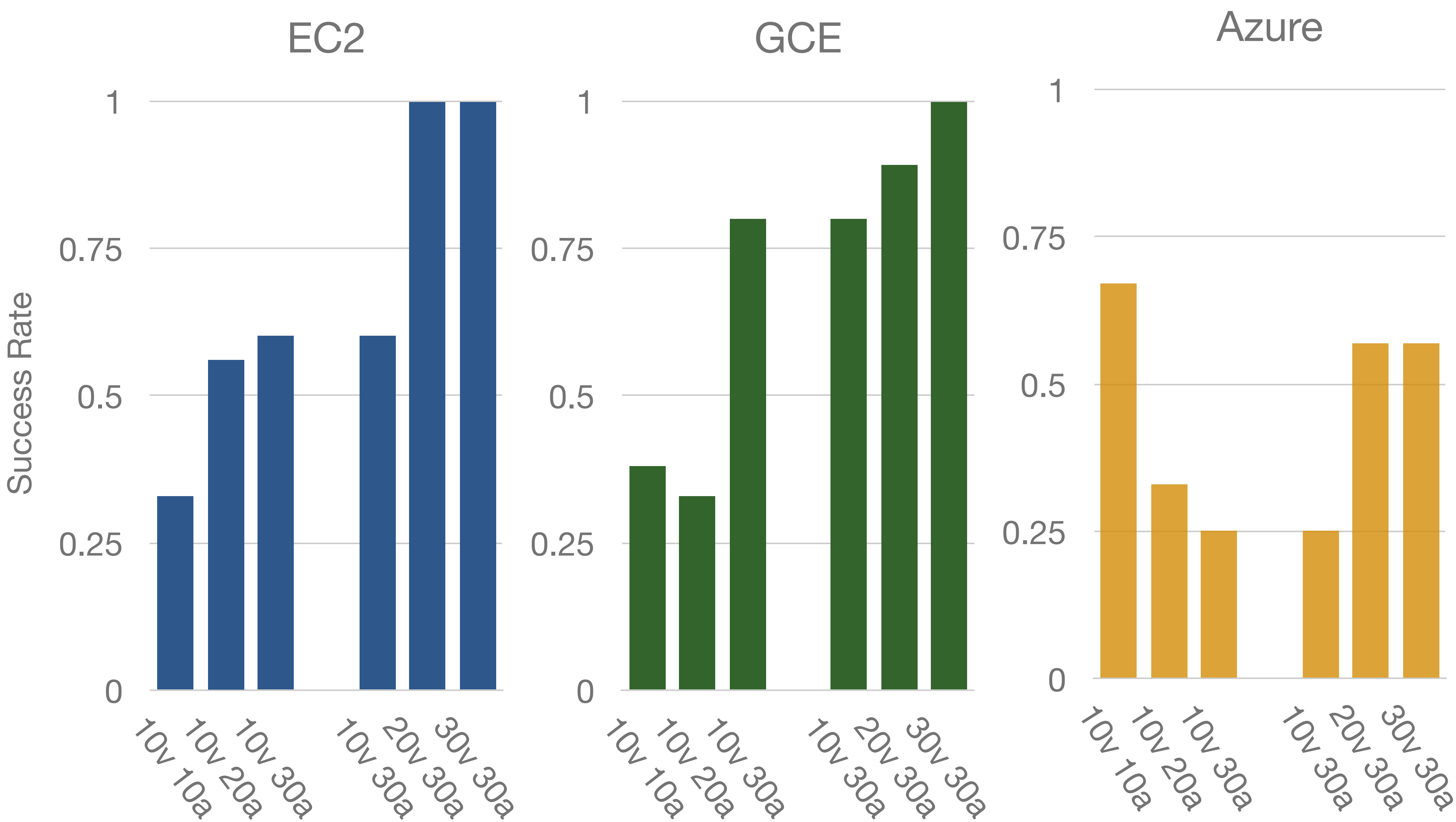
v	$a = \ln(1 - P_C)/\ln(1 - v/N)$; $P_C = 0.5$
10	**3466**

For a modest 50% success rate with 10-30 victims we need to launch ~3000 VMs

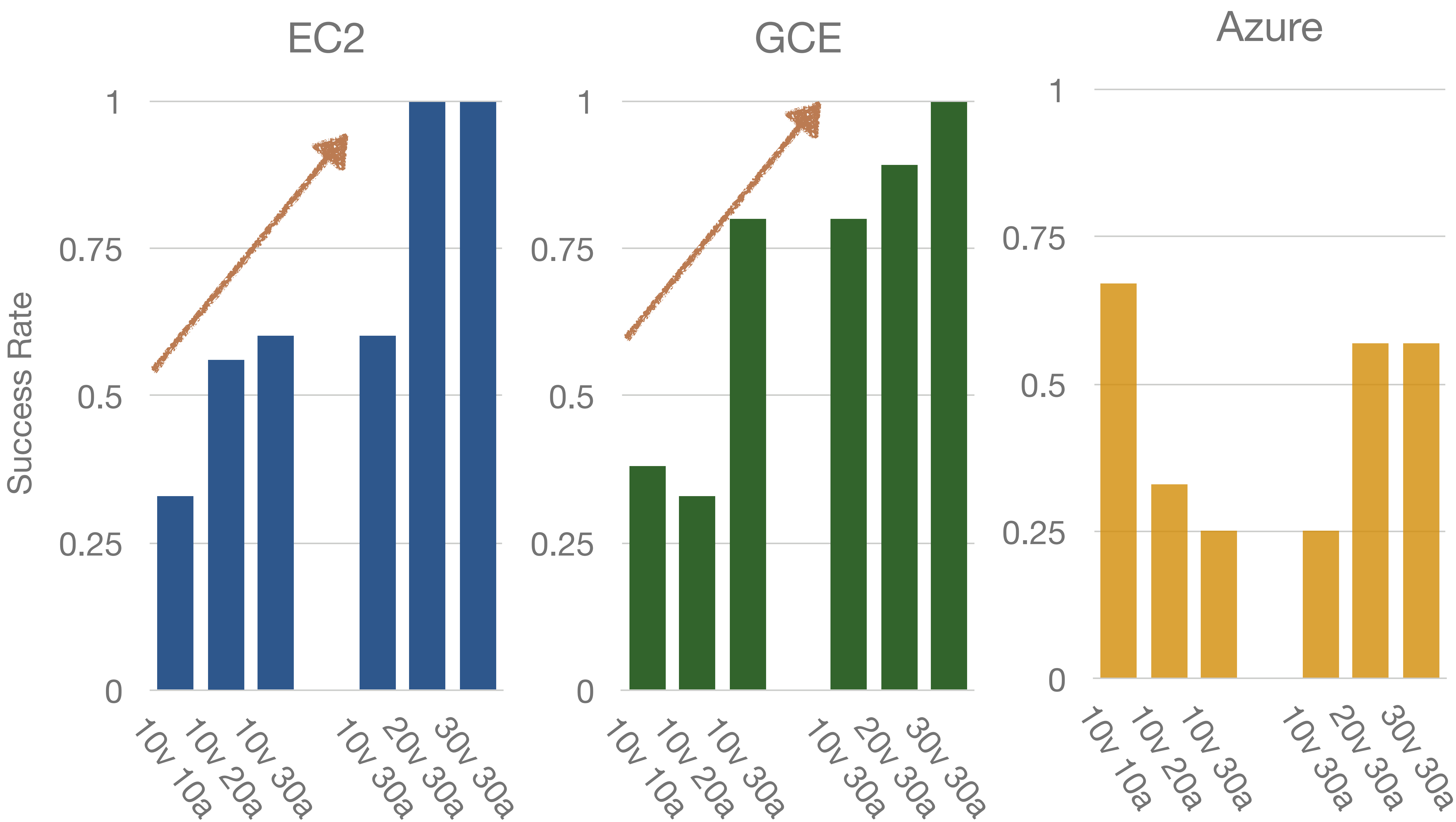
Results: Varying Number of VMs



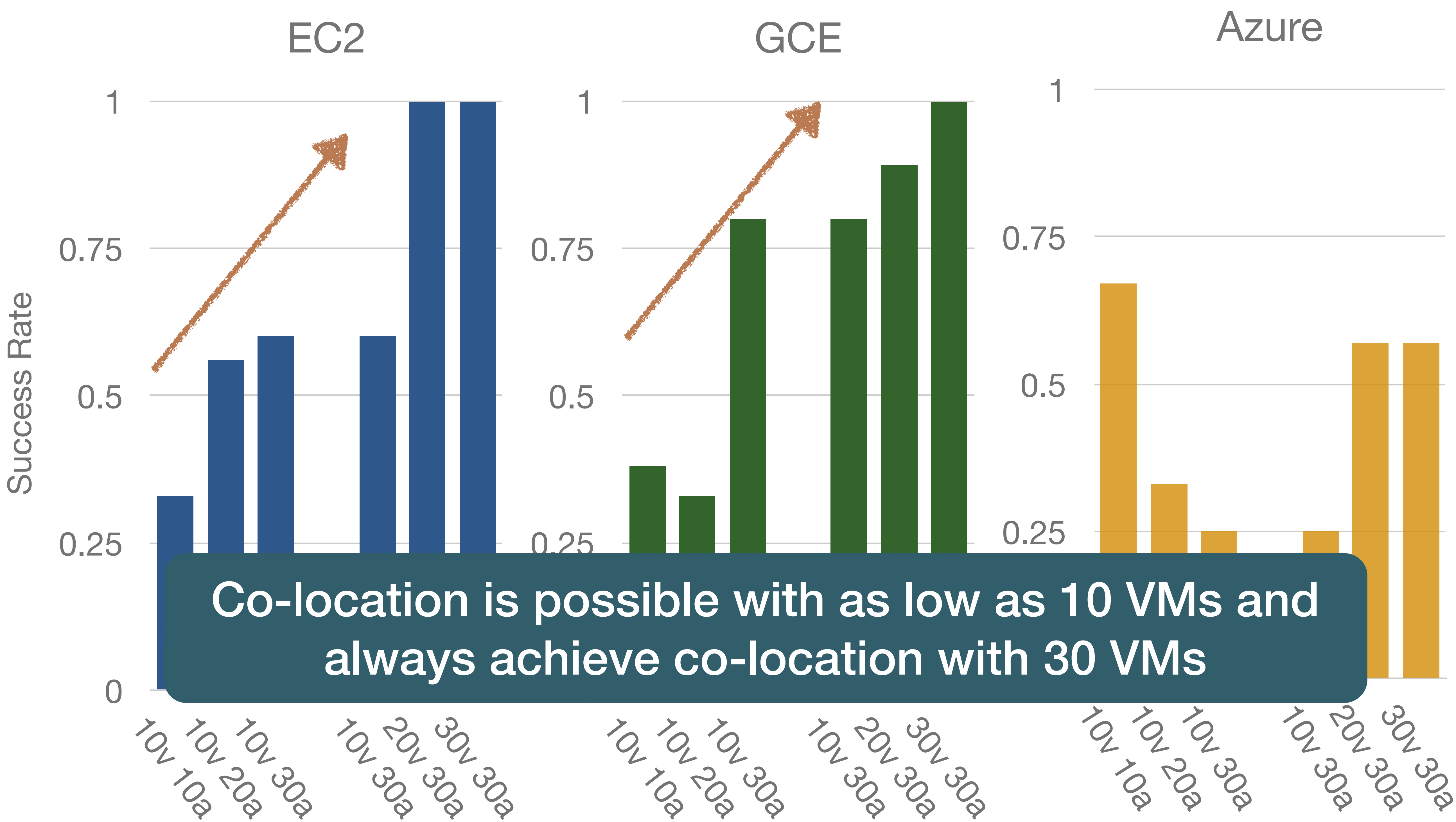
Results: Varying Number of VMs



Results: Varying Number of VMs

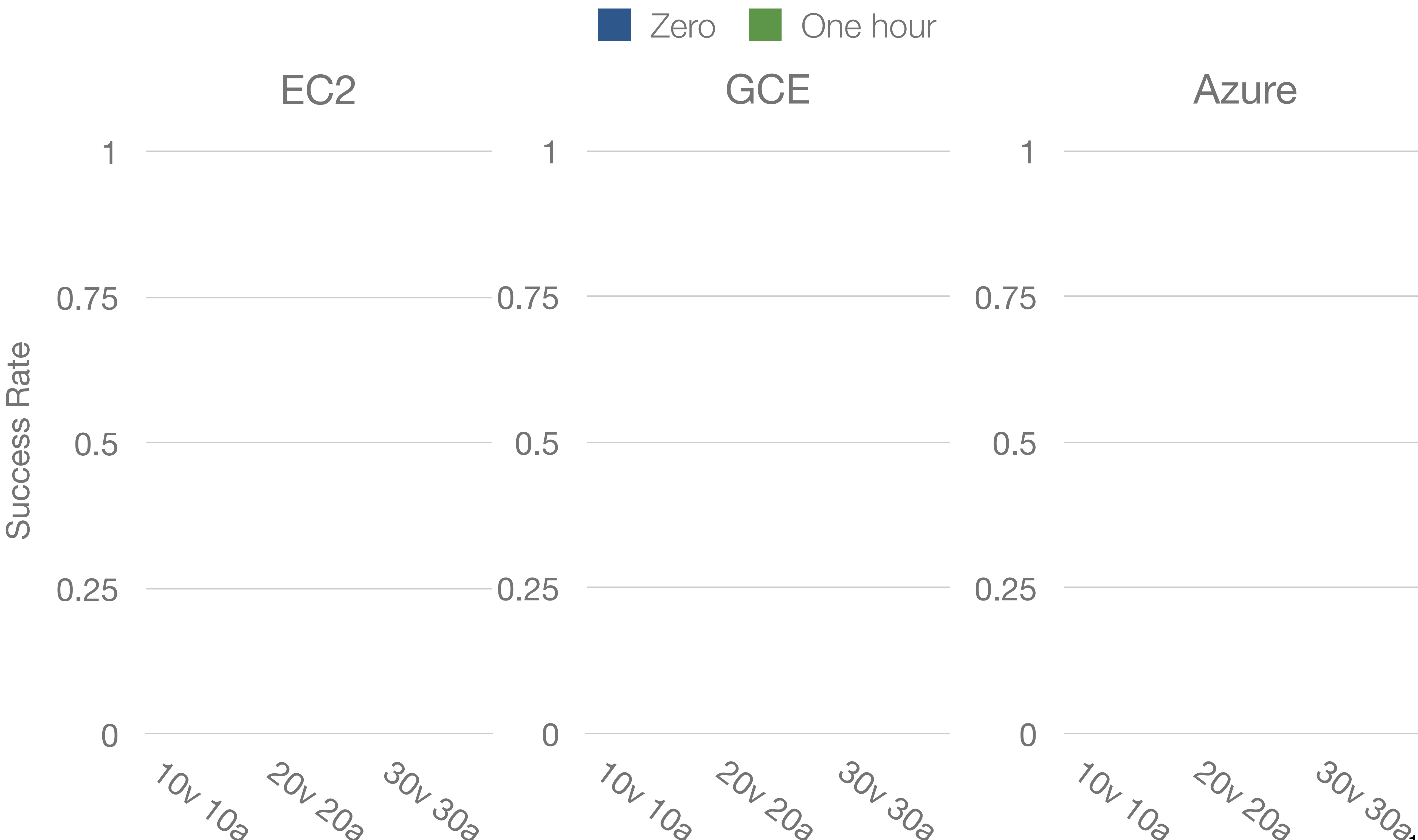


Results: Varying Number of VMs

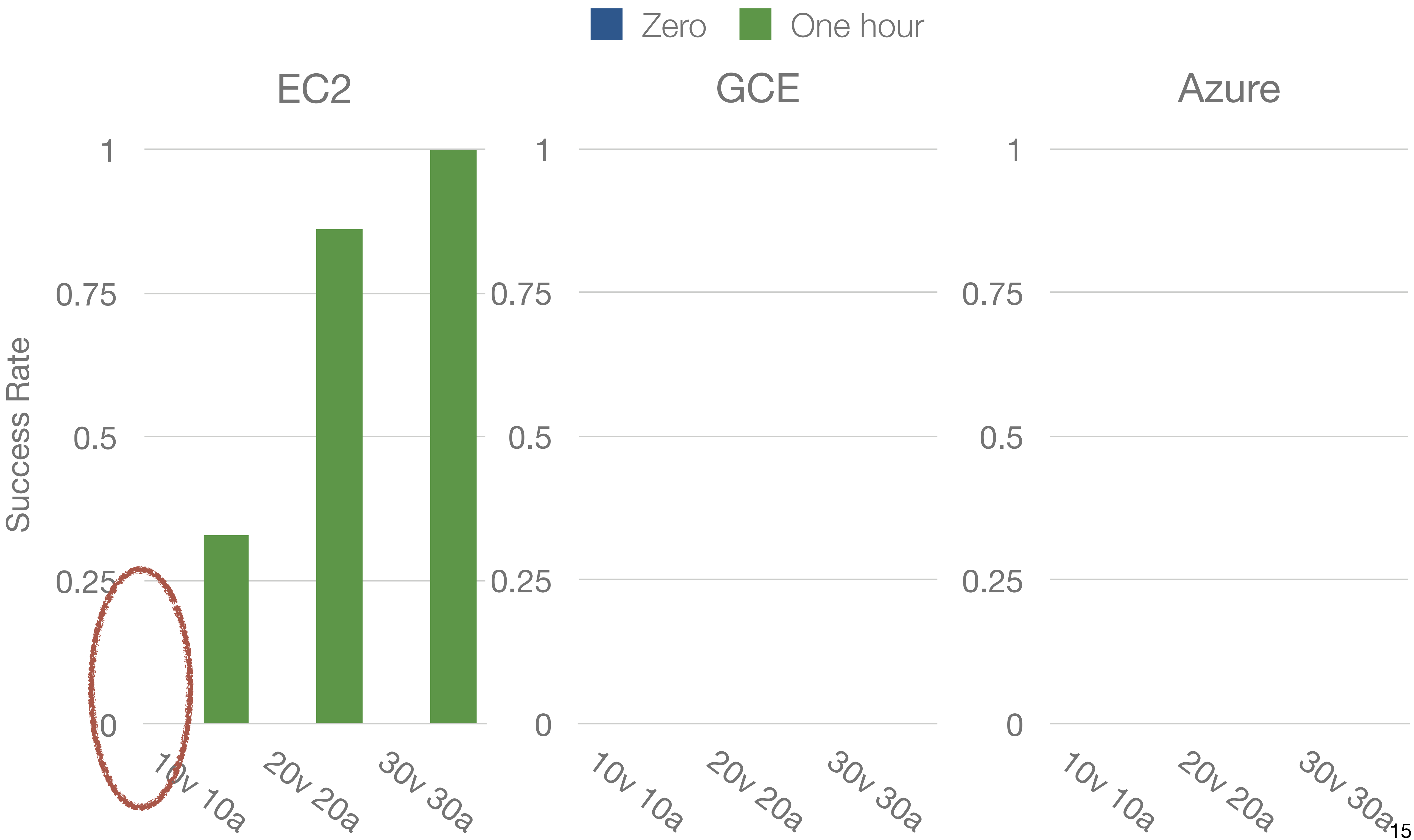


Co-location is possible with as low as 10 VMs and always achieve co-location with 30 VMs

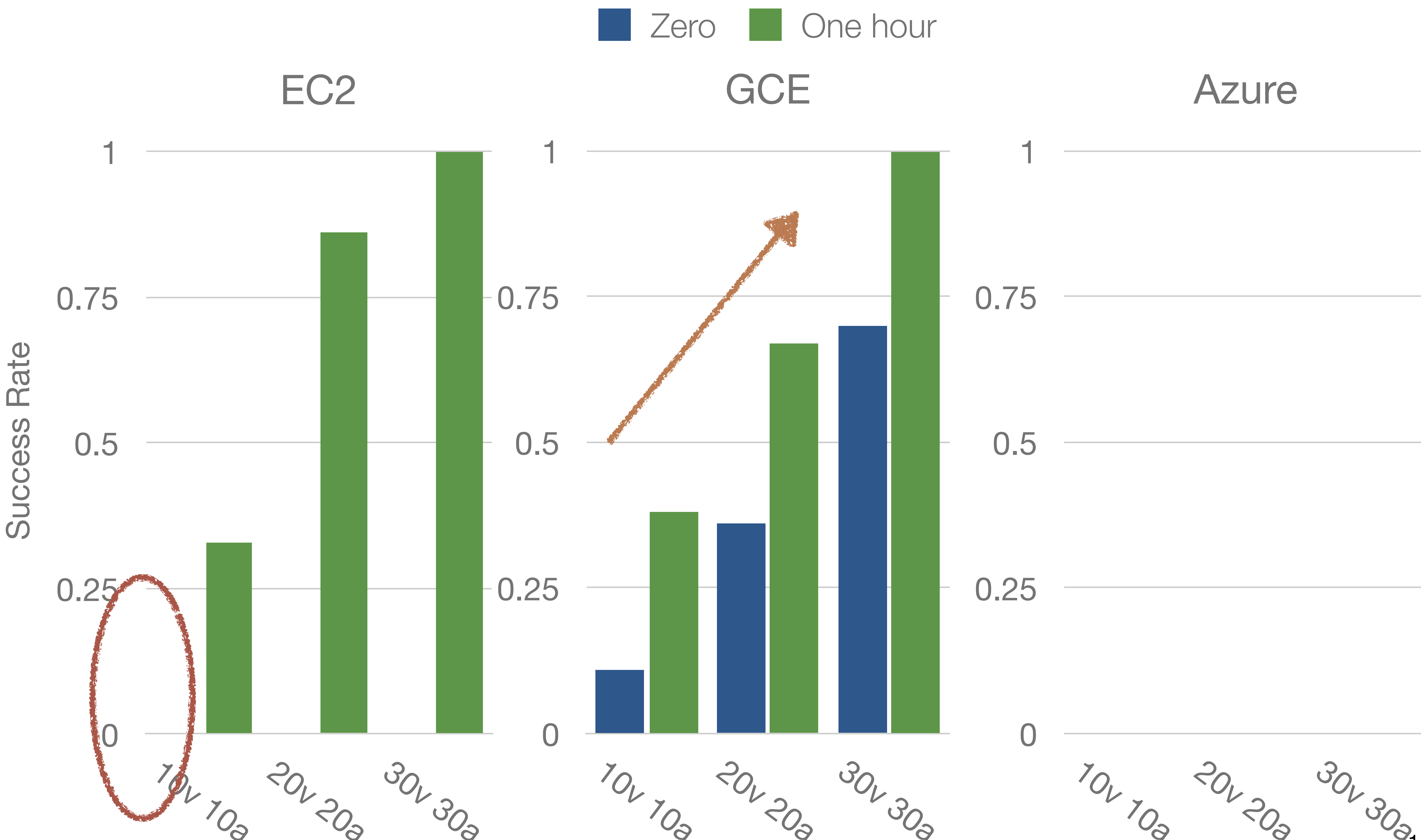
Results: Varying Delay between Launches



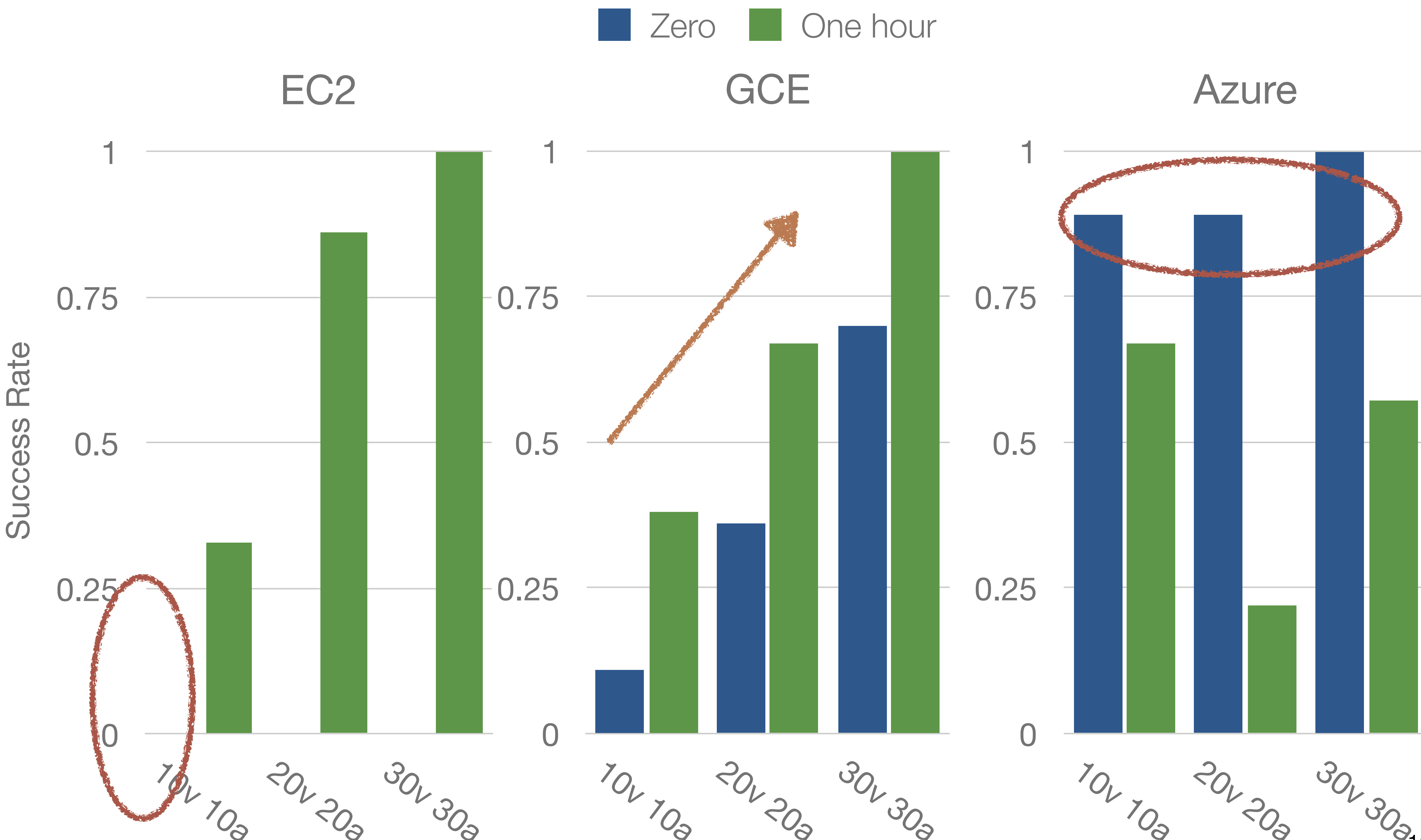
Results: Varying Delay between Launches



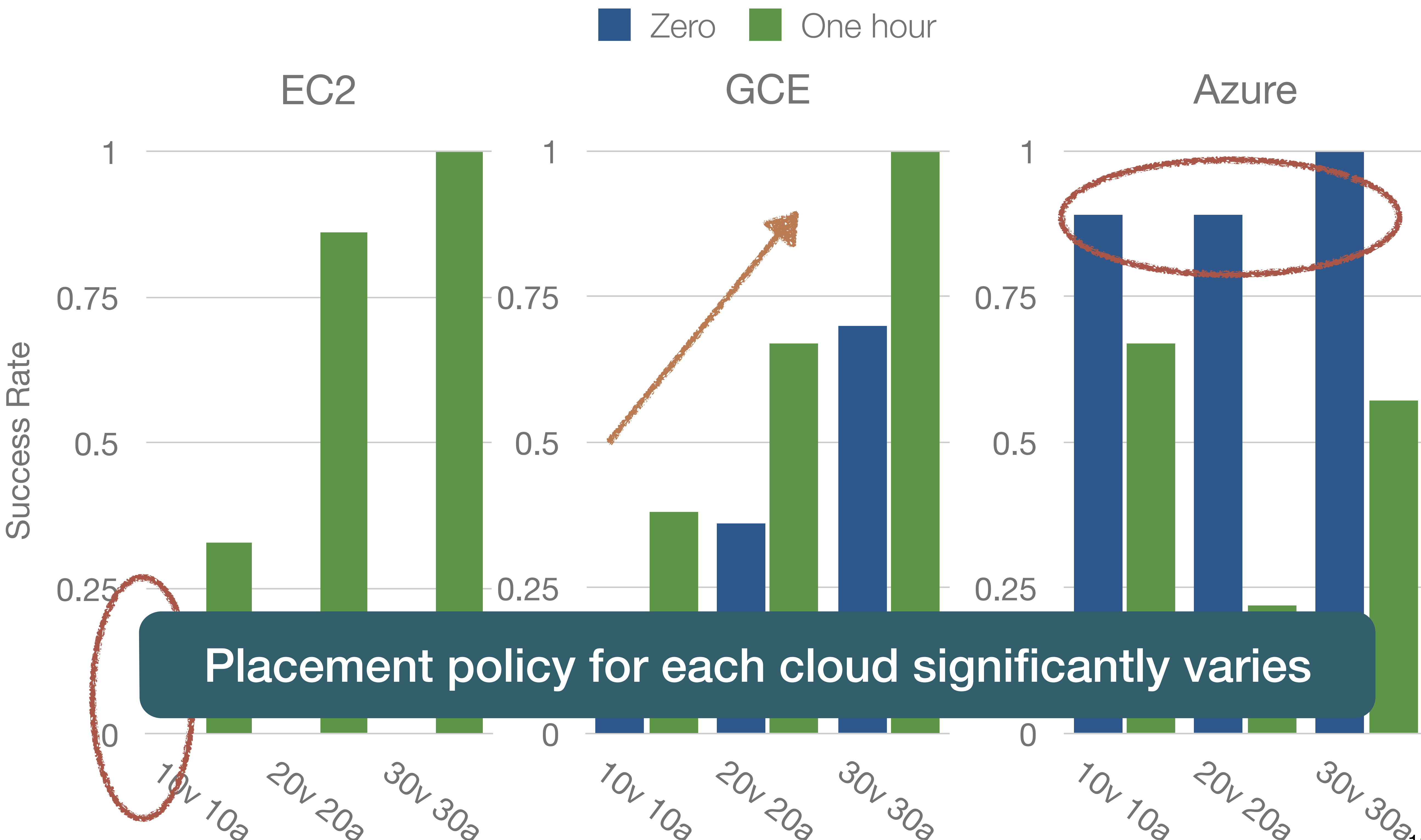
Results: Varying Delay between Launches



Results: Varying Delay between Launches

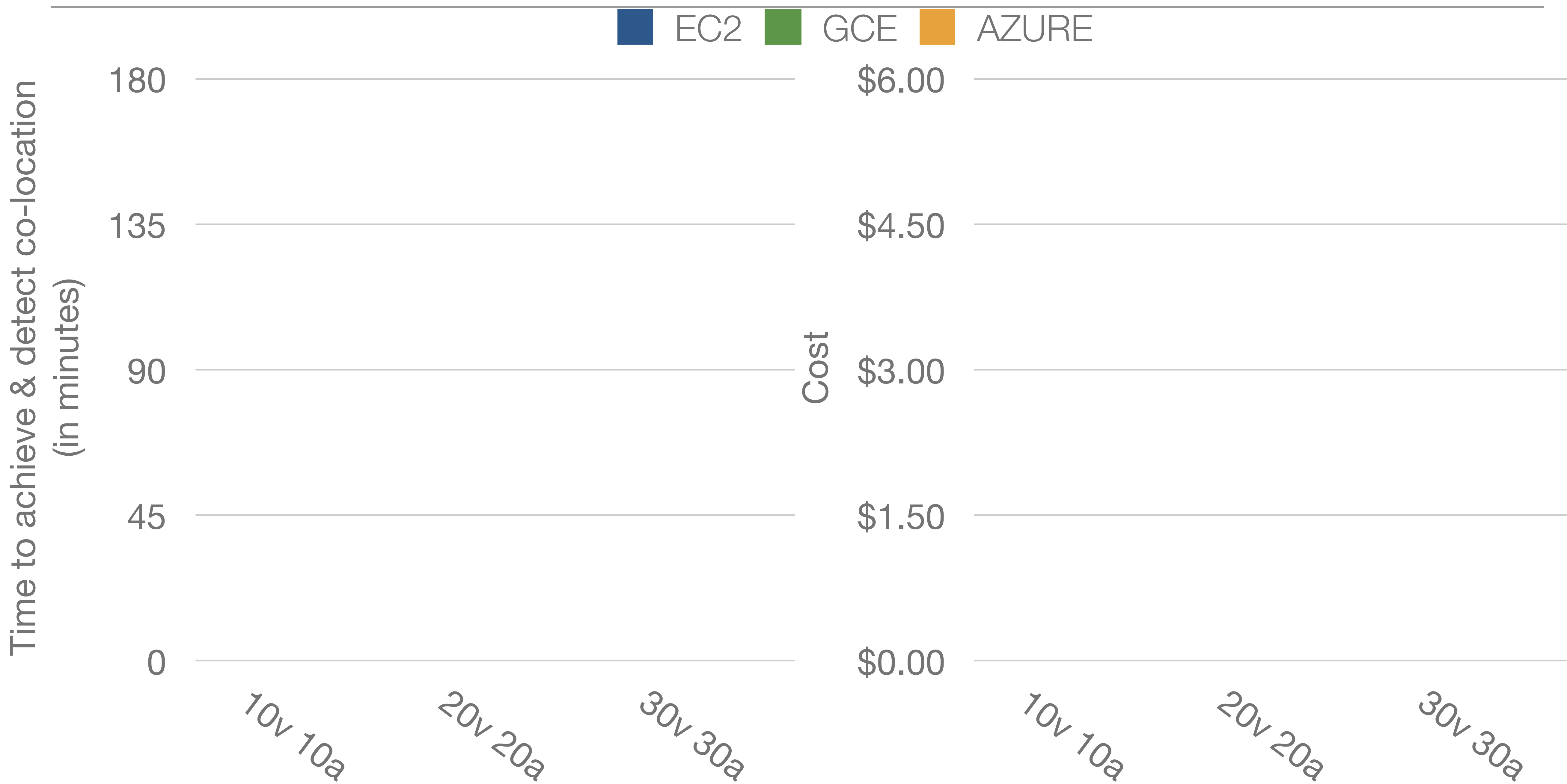


Results: Varying Delay between Launches

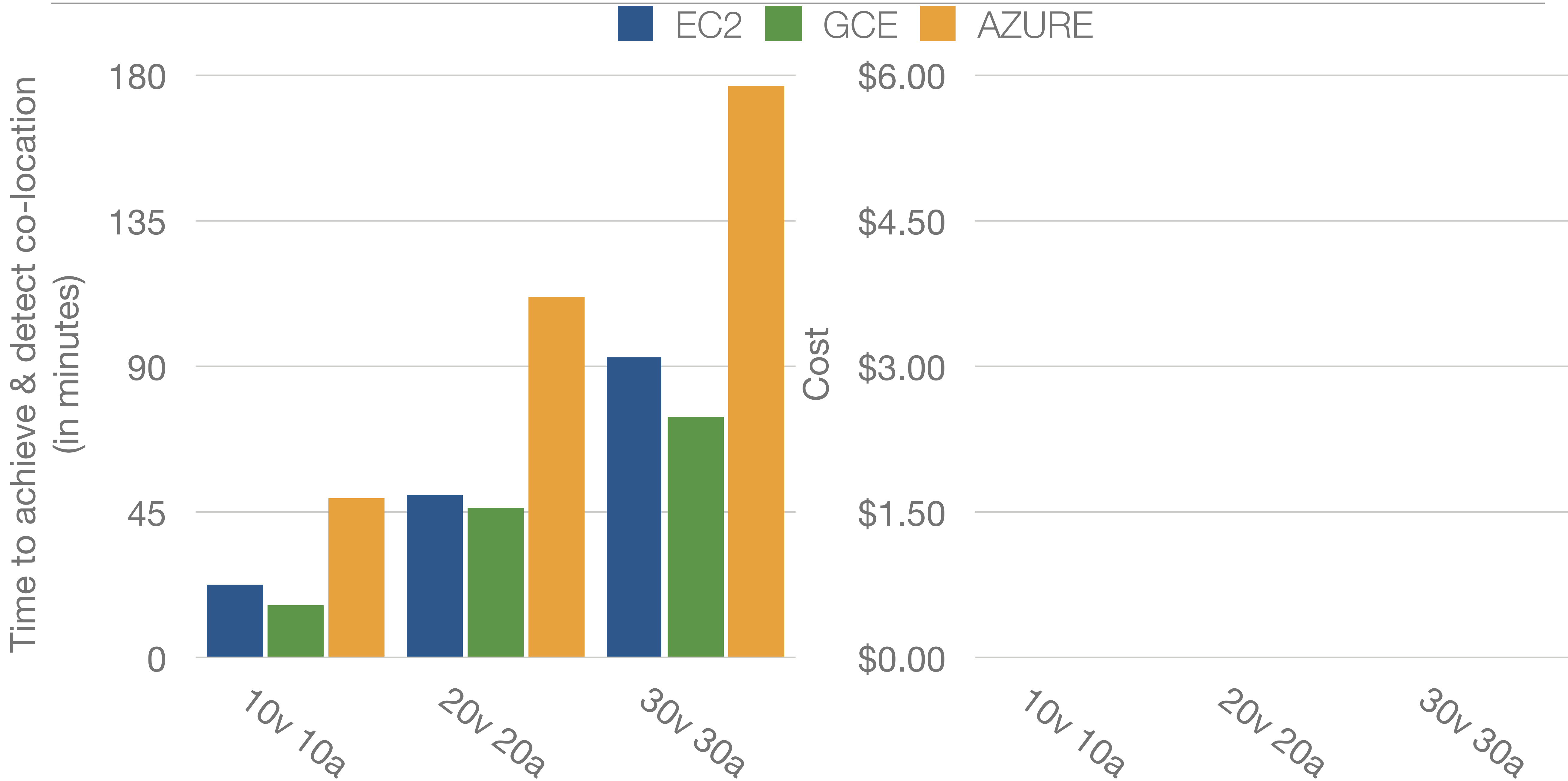


Placement policy for each cloud significantly varies

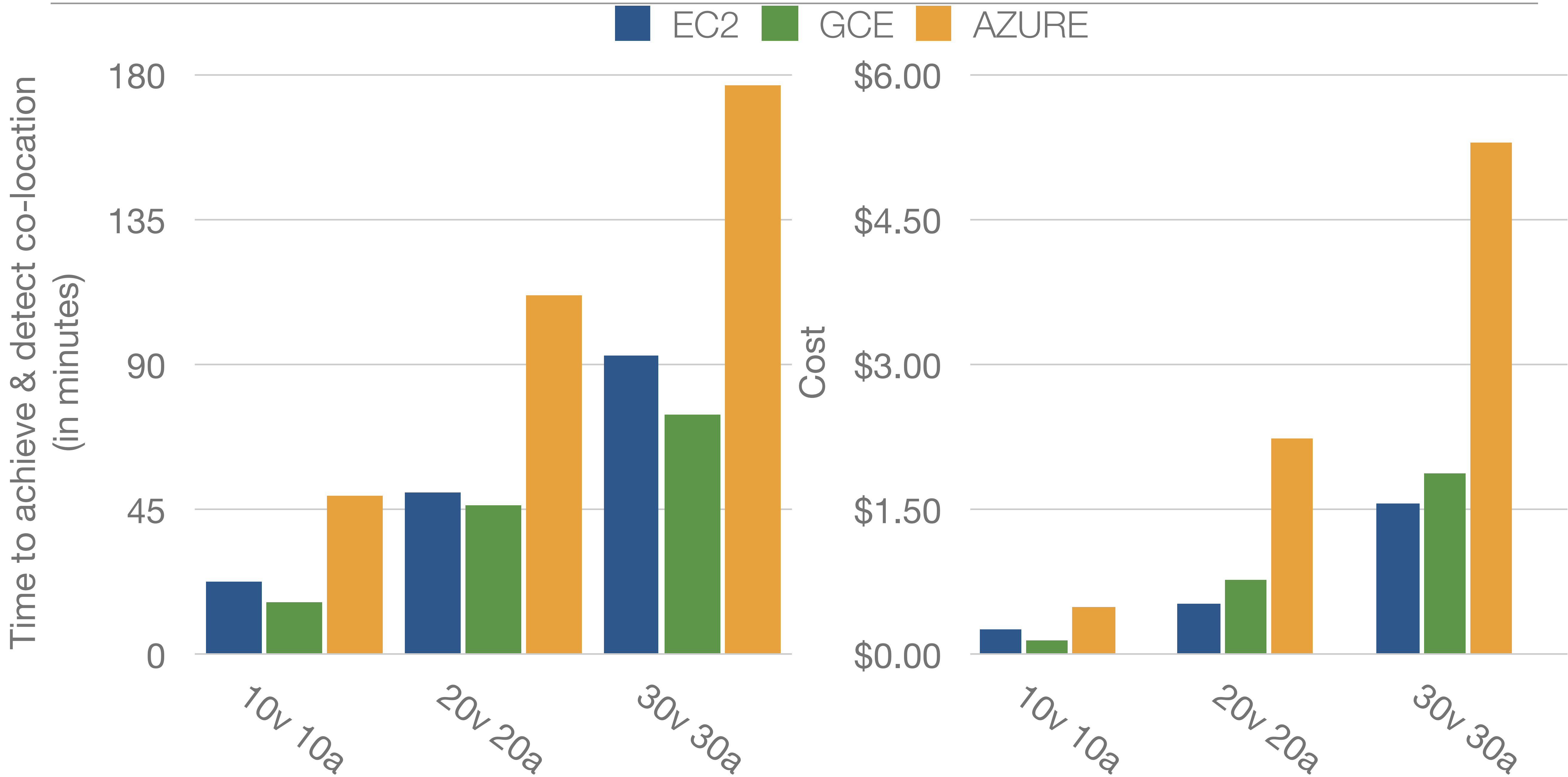
Cost of a Launch Strategy



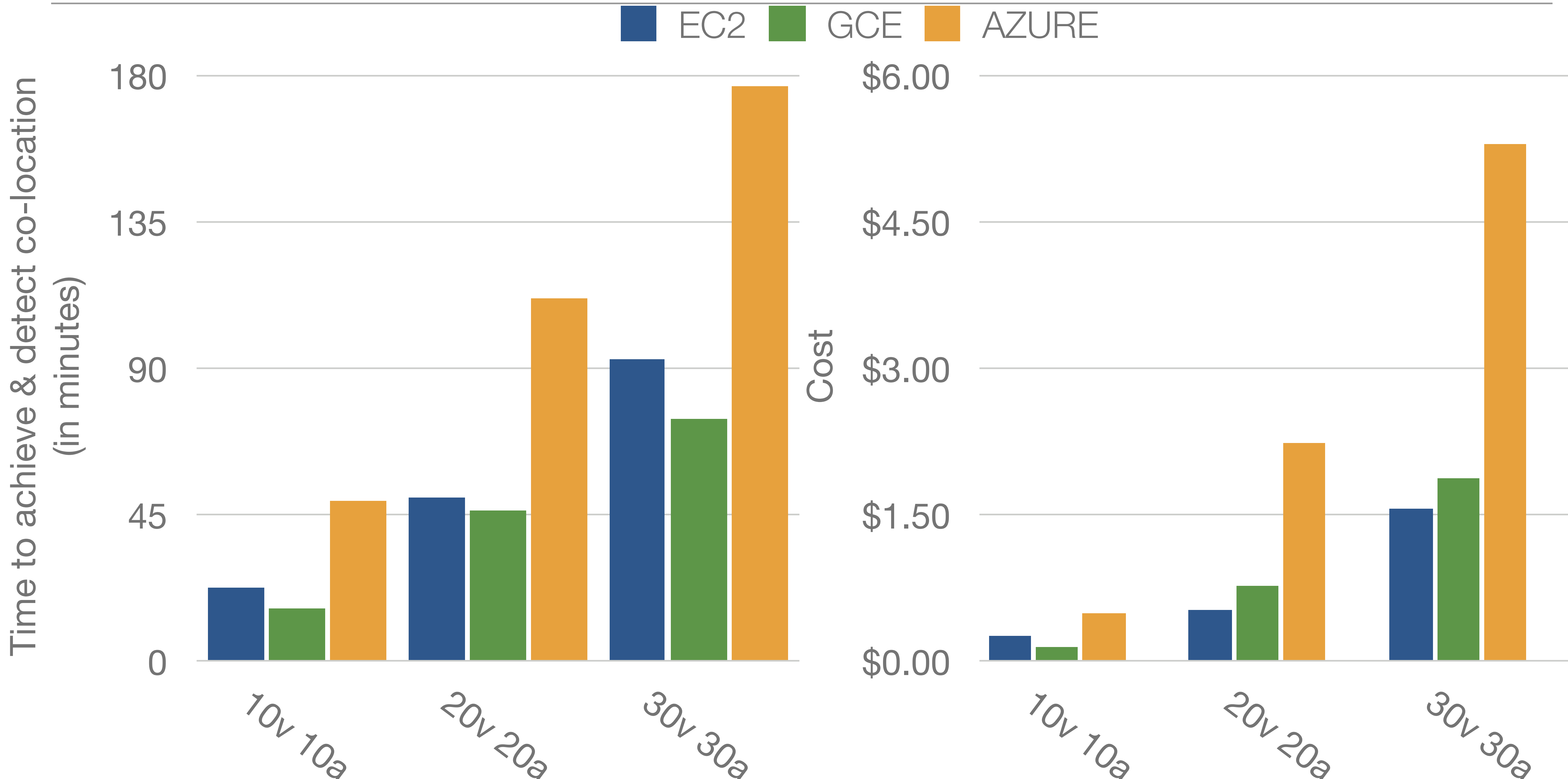
Cost of a Launch Strategy



Cost of a Launch Strategy



Cost of a Launch Strategy



The cheapest launch strategy costs as low as 14 cents

Other Interesting Results

Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)

Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
    - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.

Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
- In Azure we could co-locate 16 VMs on a single host

Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
    - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
- In Azure we could co-locate 16 VMs on a single host
- Brief experiments with platform-as-a-service, heroku

Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
    - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
- In Azure we could co-locate 16 VMs on a single host
- Brief experiments with platform-as-a-service, heroku
- … and many more in the paper

Some Strategies Work Better than Others

Some Strategies Work Better than Others

Example strategies on EC2

Launch Strategy	v x a
Launch 10 VMs in less popular datacenter	10x10
Launch 30 VMs 1 hour after victim VM launches	30x30
Launch more than 20 VMs 4 hours after victim VM launches	20x20

Some Strategies Work Better than Others

Example strategies on EC2

Launch Strategy	v x a	Cost in Cloud
Launch 10 VMs in less popular datacenter	10x10	$0.26
Launch 30 VMs 1 hour after victim VM launches	30x30	$1.56
Launch more than 20 VMs 4 hours after victim VM launches	20x20	$0.52

Some Strategies Work Better than Others

Example strategies on EC2

Launch Strategy	v x a	Cost in Cloud	Cost under Random Placement*
Launch 10 VMs in less popular datacenter	10x10	$0.26	$113.87
Launch 30 VMs 1 hour after victim VM launches	30x30	$1.56	$32.75
Launch more than 20 VMs 4 hours after victim VM launches	20x20	$0.52	$53.76

*Random Placement of VMs on N hosts,
v x a launch strategy has a probability of collision: $1 - (1 - v/N)^a$

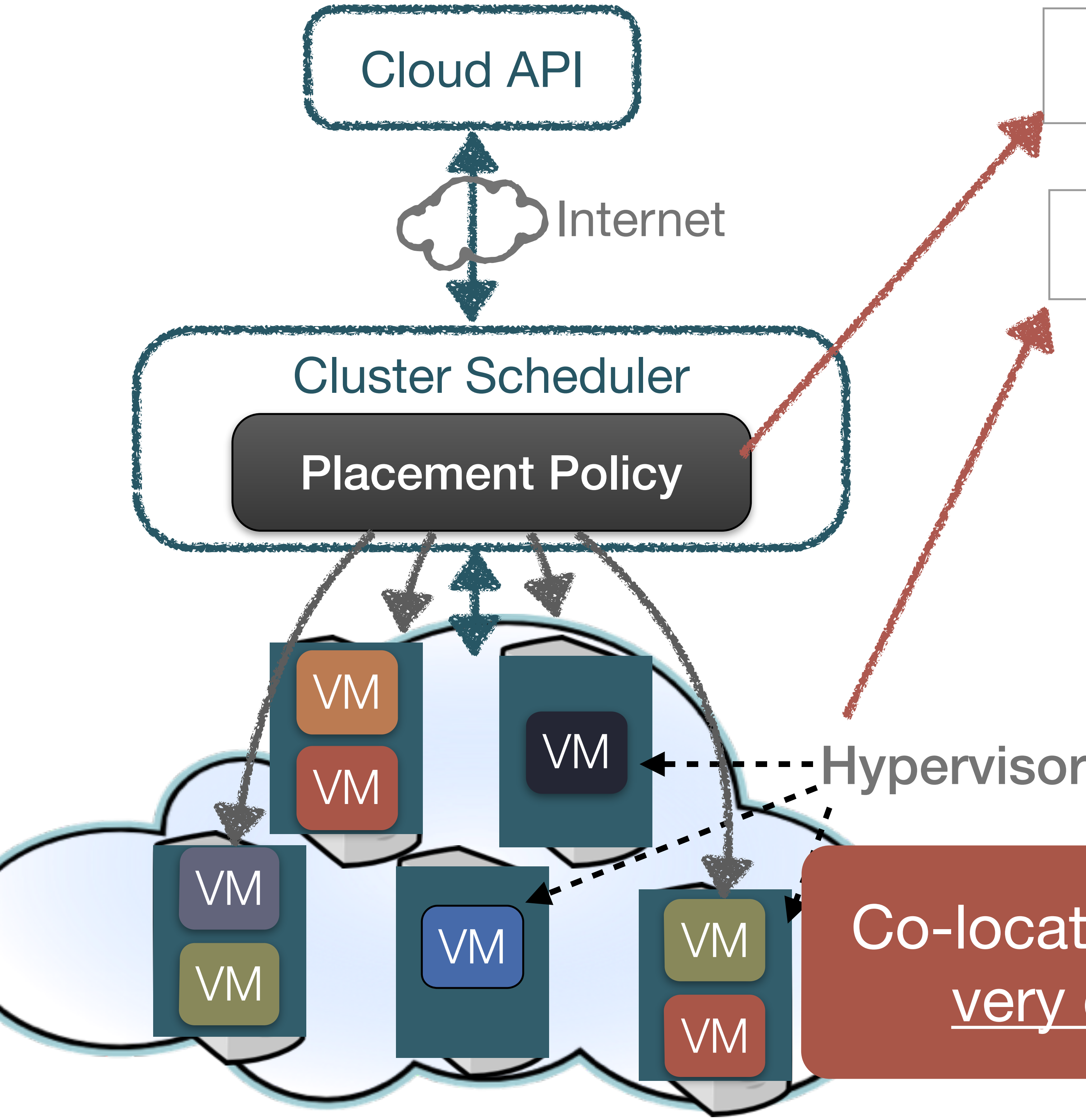
Some Strategies Work Better than Others

Example strategies on EC2

Launch Strategy	v x a	Cost in Cloud	Cost under Random Placement*	Success rate norm. w/ random*
Launch 10 VMs in less popular datacenter	10x10	$0.26	$113.87	1/0.1 (=10)
Launch 30 VMs 1 hour after victim VM launches	30x30	$1.56	$32.75	1/0.6 (=1.67)
Launch more than 20 VMs 4 hours after victim VM launches	20x20	$0.52	$53.76	1/0.33 (=3.03)

*Random Placement of VMs on N hosts,
v x a launch strategy has a probability of collision: $1 - (1 - v/N)^a$

Summary: Co-location Attacks in Modern Clouds



(1) Controls co-location

(2) Provides isolation

- **Cross-VM Side- and covert-channels** [Ristenpart'09, Xu'11, Wu'12, Zhang'12, Yarom'14, Liu'14, Gruss'15]
- **Resource-stealing, DoS** [Zhou'11, Varadarajan'12]

Co-location attacks are <u>practical</u> and <u>very cheap</u> in modern clouds!