Software at Scale

Software Architecture

Brae Webb

March 23, 2026

Question

How many concurrent users can your software handle?

Question

How many concurrent users can your software handle?

Answer

Maybe *400*? Maximum.

Question

How many concurrent users can your software handle?

Answer

Maybe *400*[1]? Maximum.

[1] HTTP server on a t2.micro EC2 instance

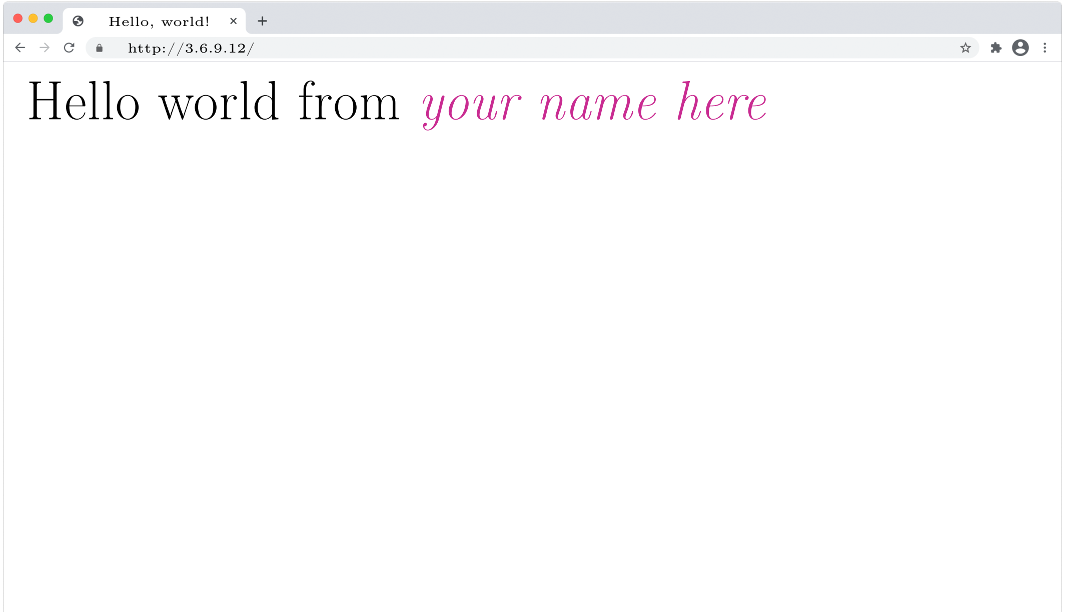
Definition 0. Stress Testing

Measure the robustness of software by pushing usage to an extreme.

Demonstration

Let’s build ‘hello world’

Our Goal

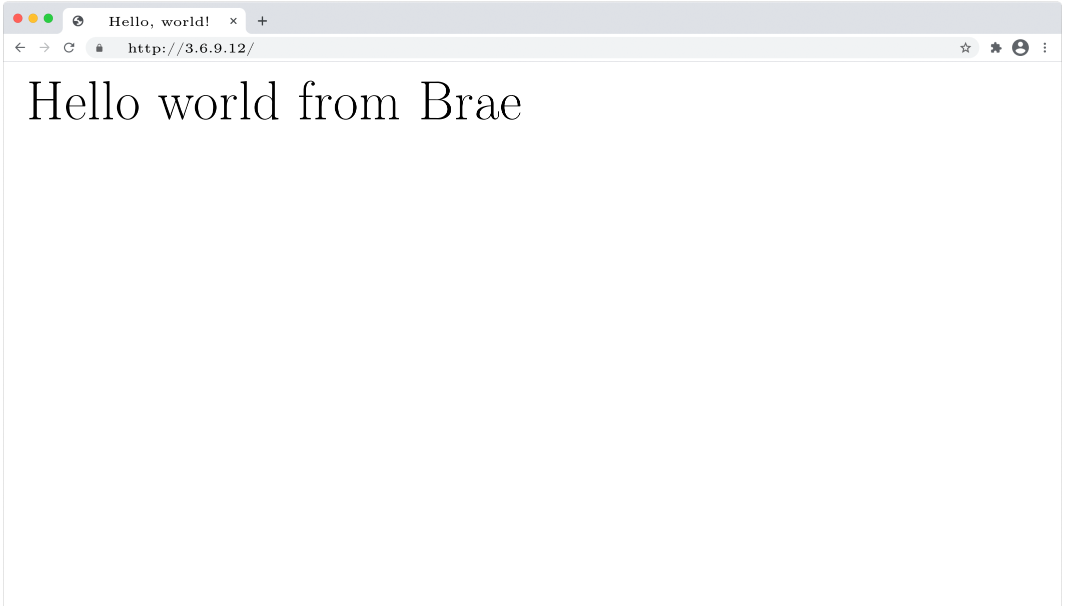


Hello, world!

http://3.6.9.12/

Hello world from *your name here*

My Goal



Hello world from Brae

```
» cat hello-server.tf
```

```
resource "aws_instance" "hello-server" {
    ami = "ami-04902260ca3d33422"
    instance_type = "t2.micro"
}
```

```
» cat hello-server.tf
```

```
resource "aws_instance" "hello-server" {
    ami = "ami-04902260ca3d33422"
    instance_type = "t2.micro"

    user_data = file("./setup.sh")
}
```

```
» cat setup.sh
```

```
#!/bin/bash
yum -y install httpd
systemctl enable httpd
systemctl start httpd
echo '<html><title>Hello, world!</title><h1>Hello world from Brae</h1></html>' > /
    var/www/html/index.html
```

```
» cat hello-server.tf
```

```
resource "aws_instance" "hello-server" {
    ami = "ami-04902260ca3d33422"
    instance_type = "t2.micro"

    user_data = file("./setup.sh")

    security_groups = [
        aws_security_group.hello-server.name
    ]
}
```

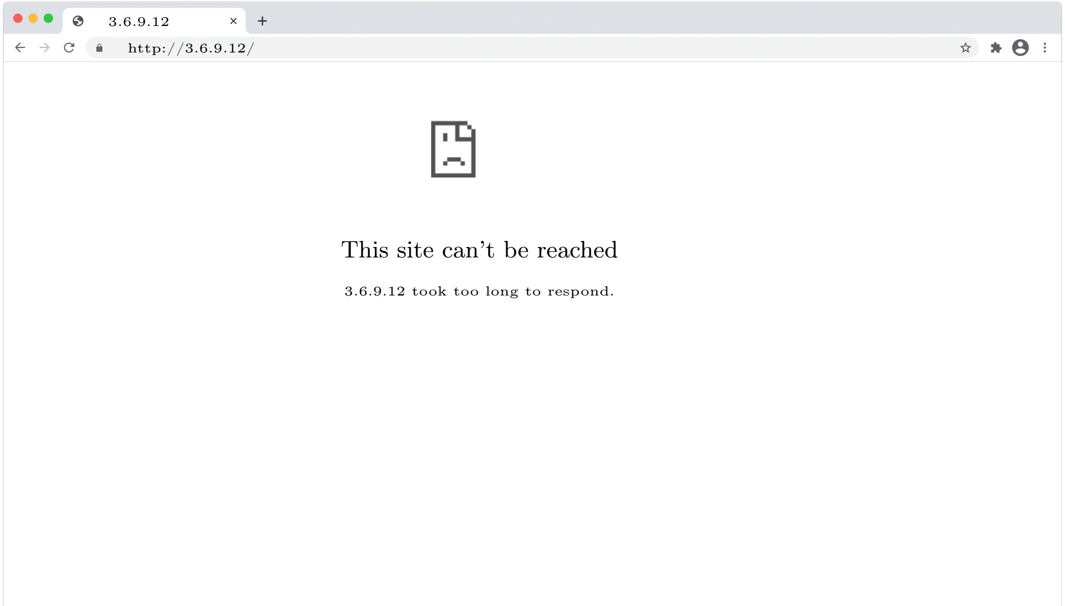
```
» cat hello-server.tf
```

```
resource "aws_instance" "hello-server" {
    ami = "ami-04902260ca3d33422"
    instance_type = "t2.micro"

    user_data = file("./setup.sh")

    security_groups = [
        aws_security_group.hello-server.name
    ]

    tags = {
        Name = "hello-server"
    }
}
```

Starting the server

```
>> terraform init
>> terraform plan
>> terraform apply
```

Before



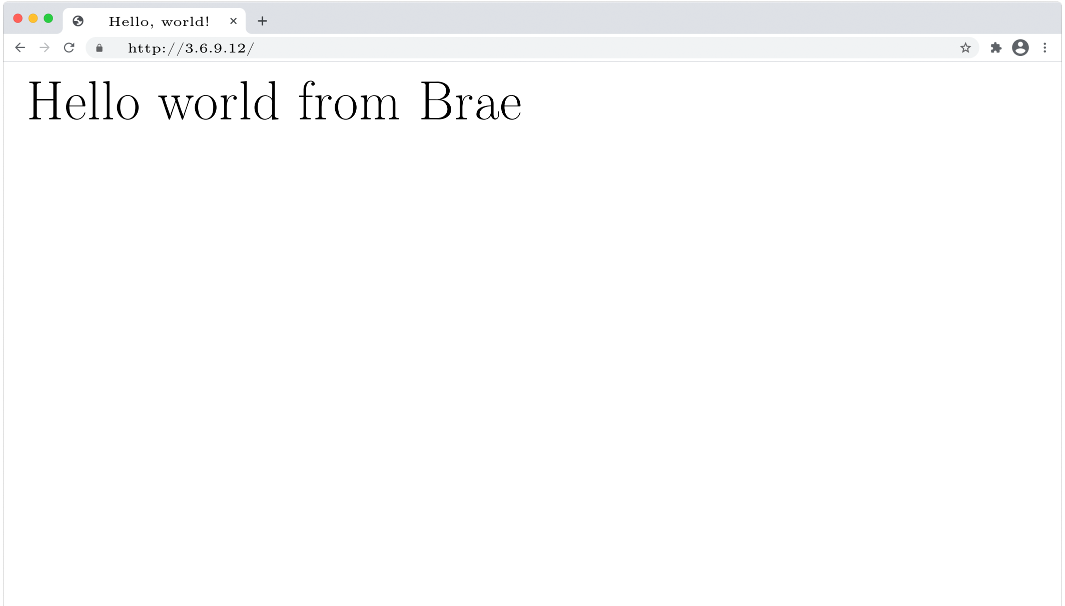
3.6.9.12

http://3.6.9.12/

This site can’t be reached

3.6.9.12 took too long to respond.

After


Hello, world!

http://3.6.9.12/

Hello world from Brae

Question

How much traffic can this website handle?

```
» cat stress-test.js
```

```
import http from 'k6/http';
import { check, sleep } from 'k6';

const IP = "http://3.6.9.12/";
export default function() {
    const res = http.get(IP);
    check(res, { 'status was 200': (r) => r.status == 200 });
    sleep(1);
}
```

```
» cat stress-test.js
```

```
import http from 'k6/http';
import { check, sleep } from 'k6';

const IP = "http://3.6.9.12/";
export const options = {
    stages: [
        { duration: '2m', target: 100 },
    ],
};
export default function() {
    const res = http.get(IP);
    check(res, { 'status was 200': (r) => r.status == 200 });
    sleep(1);
}
```

Run the tests

```
>> k6 run stress-test.js
```

Looks good so far

```
 status was 200
  100% - 347867 / 0

checks.........................: 100%
data_received..................: 100 MB 44 kB/s
data_sent......................: 27 MB 12 kB/s
iterations.....................: 347997 152.552084/s
vus............................: 1 min=1 max=400
```

Let’s upgrade the traffic

```
» cat stress-test.js
```

```
export const options = {
    stages: [
        { duration: '2m', target: 100 },
        { duration: '5m', target: 100 },
        { duration: '2m', target: 200 },
        { duration: '5m', target: 200 },
        { duration: '2m', target: 300 }, // around the breaking point
        { duration: '5m', target: 300 },
        { duration: '2m', target: 400 }, // beyond the breaking point
        { duration: '5m', target: 400 },
        { duration: '2m', target: 0 }, // scale down
    ],
};
```

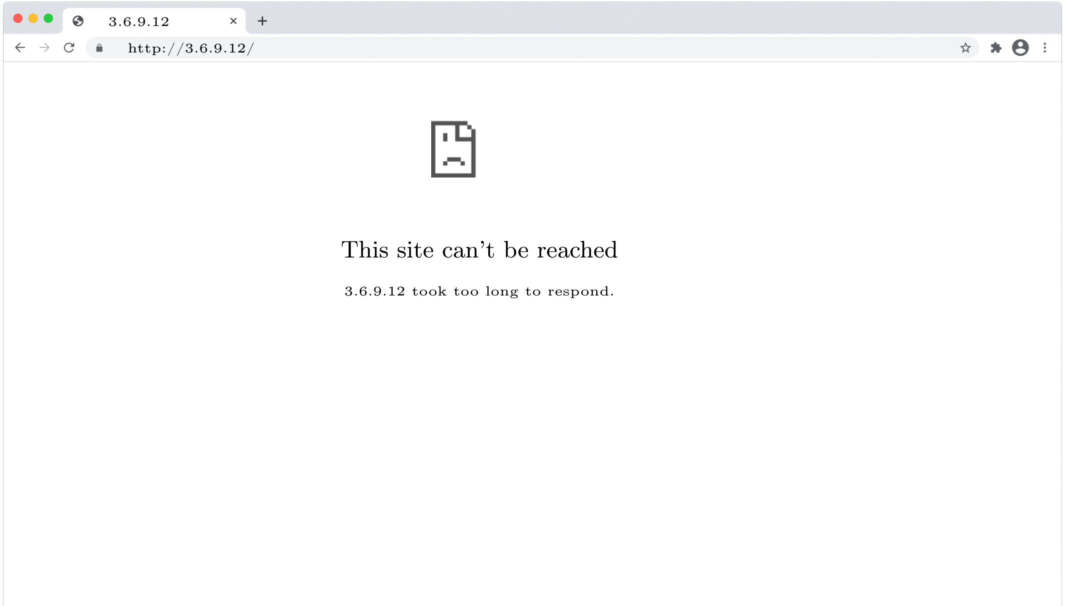
And run the tests again

```
>> k6 run stress-test.js
```

Oh no...

```
status was 200
99% - 347867 / 130

checks.........................: 99.96%
data_received..................: 100 MB 44 kB/s
data_sent......................: 27 MB 12 kB/s
iterations.....................: 347997 152.552084/s
vus............................: 1 min=1 max=400
```

Back to square one


3.6.9.12

http://3.6.9.12/

This site can’t be reached

3.6.9.12 took too long to respond.

Question

How can we fix this?

Question

How can we fix this?

Answer

More servers?

```
» cat hello-server.tf
```

```
resource "aws_instance" "hello-server" {
    ami = "ami-04902260ca3d33422"
    instance_type = "t2.micro"

    user_data = file("./setup.sh")

    security_groups = [
        aws_security_group.hello-server.name
    ]

    tags = {
        Name = "hello-server"
    }
}
```

```
» cat hello-scale.tf
```

```
resource "aws_instance" "hello-server" {
  count = 4

  ami = "ami-04902260ca3d33422"
  instance_type = "t2.micro"
  user_data = file("${path.module}/setup.sh")

  security_groups = [
    aws_security_group.hello-server.name
  ]

  tags = {
    Name = "hello-server-${count.index}"
  }
}
```

Definition 0. Target Group

A collection of EC2 instances.

More specifically, a collection of network connection points to EC2 instances.

An empty HTTP target group

```
» cat hello-scale.tf
```

```
resource "aws_lb_target_group" "hello-target" {
  name = "hello-target-group"
  port = 80
  protocol = "HTTP"
  vpc_id = aws_security_group.hello-server.vpc_id
}
```

Definition 0. Health Check

Monitors attributes of hardware or software to detect deficiencies.

Add a health check

```
» cat hello-scale.tf
```

```
resource "aws_lb_target_group" "hello-target" {
  name = "hello-target-group"
  port = 80
  protocol = "HTTP"
  vpc_id = aws_security_group.hello-server.vpc_id

  health_check {
    port = 80
    protocol = "HTTP"
    timeout = 5
    interval = 10
  }
}
```

Add our instances to the target group

```
» cat hello-scale.tf
```

```
resource "aws_lb_target_group_attachment" "hello-target-link" {
  count = length(aws_instance.hello-server)
  target_group_arn = aws_lb_target_group.hello-target.arn
  target_id = aws_instance.hello-server[count.index].id
  port = 80
}
```

Definition 0. Load Balancer

A networking tool to route and distribute traffic to targets.

Create a load balancer

```
» cat hello-scale.tf
```

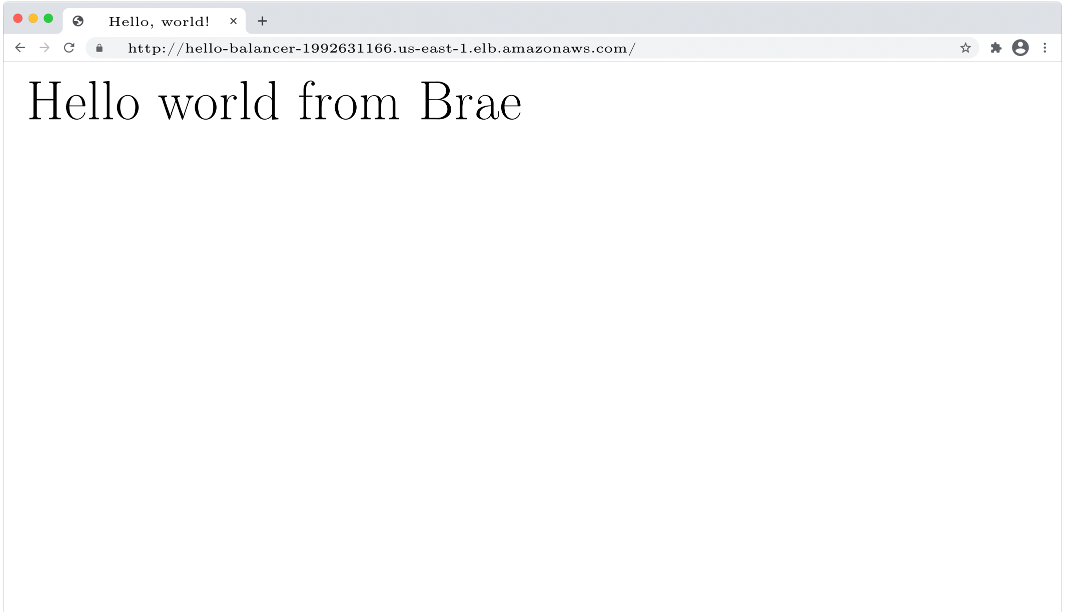
```
data "aws_subnet_ids" "nets" {
    vpc_id = aws_security_group.hello-server.vpc_id
}

resource "aws_lb" "hello-balancer" {
  name = "hello-balancer"
  internal = false
  load_balancer_type = "application"
  subnets = aws_subnet_ids.nets.ids
  security_groups = [
    aws_security_group.hello-server.name
  ]
}
```

Route load balancer traffic to the target group

```
» cat hello-scale.tf
```

```
resource "aws_lb_listener" "app" {
  load_balancer_arn = aws_lb.hello-balancer.arn
  port = "80"
  protocol = "HTTP"

  default_action {
    type = "forward"
    target_group_arn = aws_lb_target_group.hello-target.arn
  }
}
```

We're live!



Hello, world!

http://hello-balancer-1992631166.us-east-1.elb.amazonaws.com/

Hello world from Brae

Exercise

Use *k6* to determine the new *load limits*