# Content Distribution Network with Web Application Firewall Sprint #5

## (CDN with WAF)

**Team Member:**
Anand Sanmukhani
Berk Gur
Hao "Edward" Xu
Samit "Jade" Dhangwattanotai
Xuanhao Mi

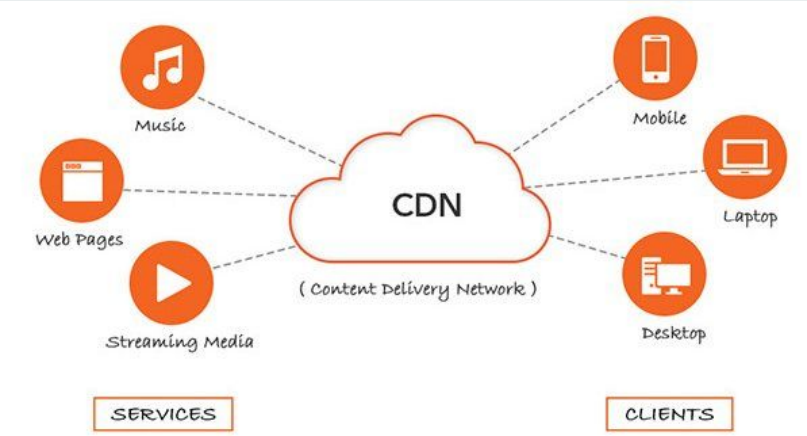
BOSTON UNIVERSITY

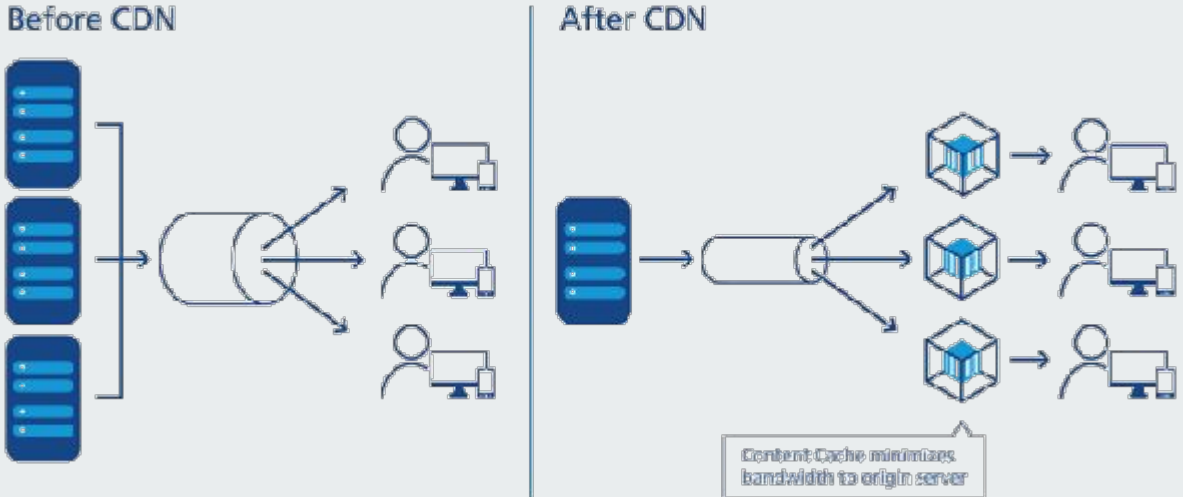
Project Recap:

- **Web Server:** Build content distribution network which helps improve efficiency of web access by the use of cache servers.
- **CDN**: distribute service among large number of servers. reducing bandwidth costs
  - **DNS Server**
  - **Data Store**
  - **Varnish Cache Servers with Web Application Firewall:** web application accelerator also known as a caching HTTP reverse proxy.

BOSTON UNIVERSITY

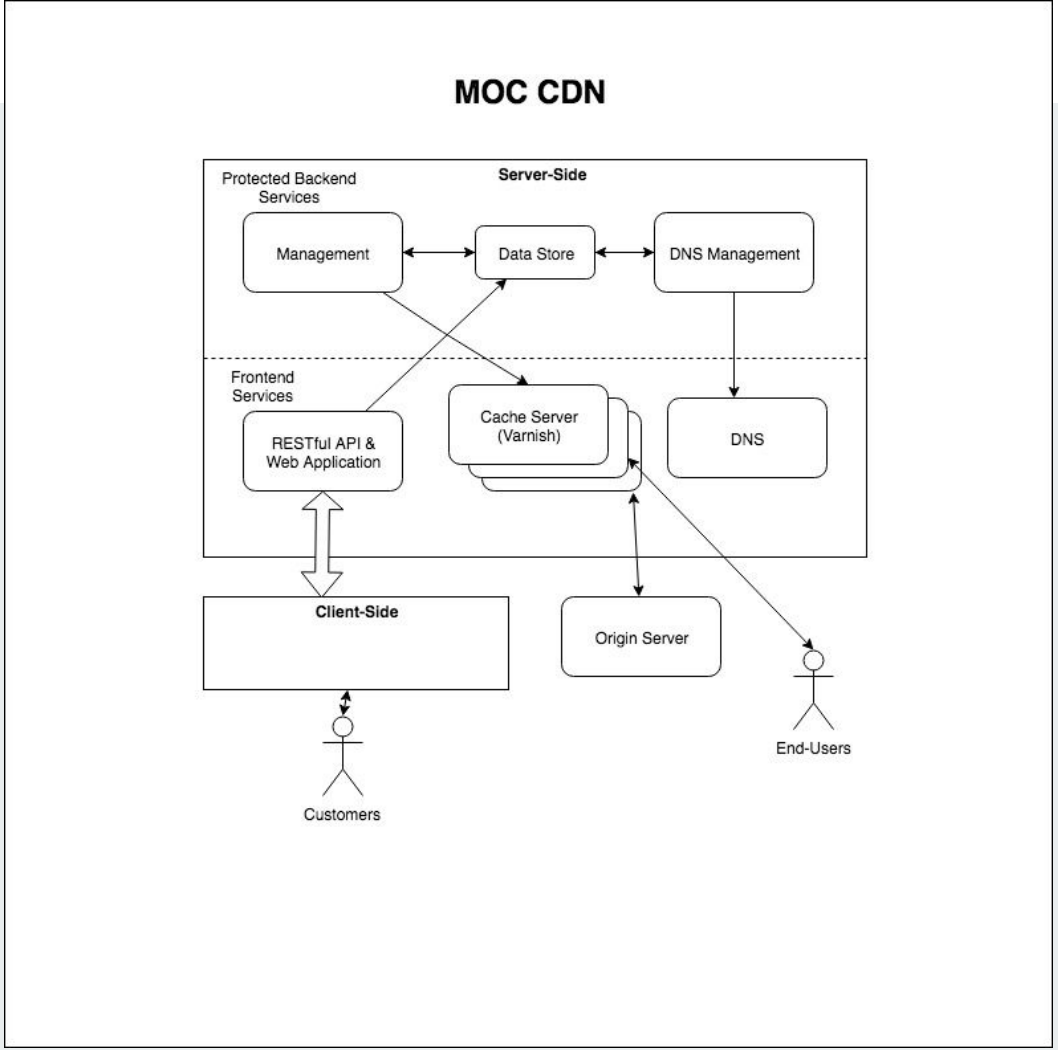
# What is a CDN? (Content Distribution Network)



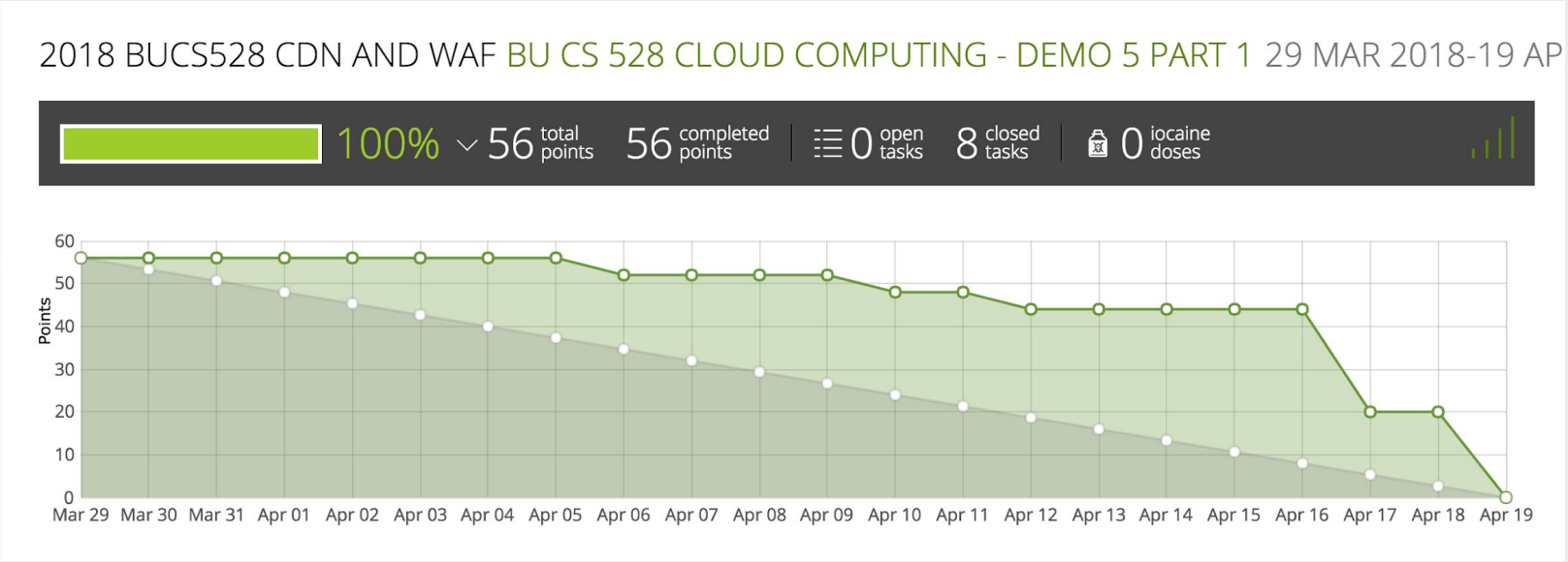

BOSTON UNIVERSITY

# Project Architecture




BOSTON UNIVERSITY

# Burn down chart




BOSTON UNIVERSITY

# Varnish Security Firewall

VSF aims to provide:

- A standardized framework for security-related filters
- Several core rule-sets
- A limited set of default 'handlers', for instance CGI scripts to call upon when Bad Stuff happens.

BOSTON UNIVERSITY

```
 GNU nano 2.5.3                          File: handlers.vcl

    #call sec_drop;      # 805  # drop the request (not implemented)
    #call sec_myhandler; # any  # do your own thing (as below)
}

# Here you can specify what gets logged when a rule triggers.
sub sec_log {
    std.log("security.vcl alert xid:" + req.xid + " " + req.proto
        + " [" + req.http.X-VSF-Module + "-" + req.http.X-VSF-RuleID + "]"
        + req.http.X-VSF-Client
        + " (" +  req.http.X-VSF-RuleName + ") ");
    #std.syslog(6, "<VSF> " + std.time2real(now) + " [" + req.http.X-VSF-RuleName + "/ruleid:" + req.http.X-VSF-RuleID + "]: $
}


/* You can define your own handlers here if you know a little vcl.
 * The default handlers are defined in main.vcl
 * remember that it must be referenced in the code above */

/* sample handler, contains sample code for all handler types */
sub sec_myhandler {
    # perform an action based on the error code as above.

    return (synth(800, "Blahblah")); # debug response

    set req.http.X-VSF-Response = "we don't like your kind around here";
    return (synth(801, "Rejected"));

    set req.http.X-VSF-Response = "http://u.rdir.it/hit/me/please";
    return (synth(802, "Redirect"));

    # send to sec_honey backend
    return (synth(803, "Honeypot me"));

    set req.http.X-VSF-Response = "<h1>Whatever</h1> so you think you can dance?";
    return (synth(804, "Synthesize"));

    return (synth(805, "Drop"));
}
```

# Progress in sprint 5

1. Configured Varnish Security Firewall
2. Tested VSF event with log
3. Developed auto-scaling
4. Kept integrating system

Problems encountered:

- Cannot add additional user to MOC and cannot reset password for existing users
- VSF implemented: syntax is correct, although varnishlog command freezes
- Connection error. Could not connect to db server: TCP/IP connections on port 5432 (Postgres Database default port)

BOSTON UNIVERSITY

# Future Plan

Final Sprint Objectives

- Finish development the Data Store instance on MOC with autoscaling
  - Using MOC API to spin up new instances
  - Duplicate Varnish server. Need solve for authorization key.
- Finish implementation VSF on Varnish Cache Server
  - Display event log of errors caught by VSF
- Integrate the whole system

Extended goal

- Update load balancing from round robin to priority queue

BOSTON UNIVERSITY

# Demo

# Thank you, Questions?