Hello, Galaxy!

Hello World at Scale

Dylan Beattie

Progressive.NET Tutorials 2017

@dylanbeattie #prognet

Dylan Beattie *(me!)*

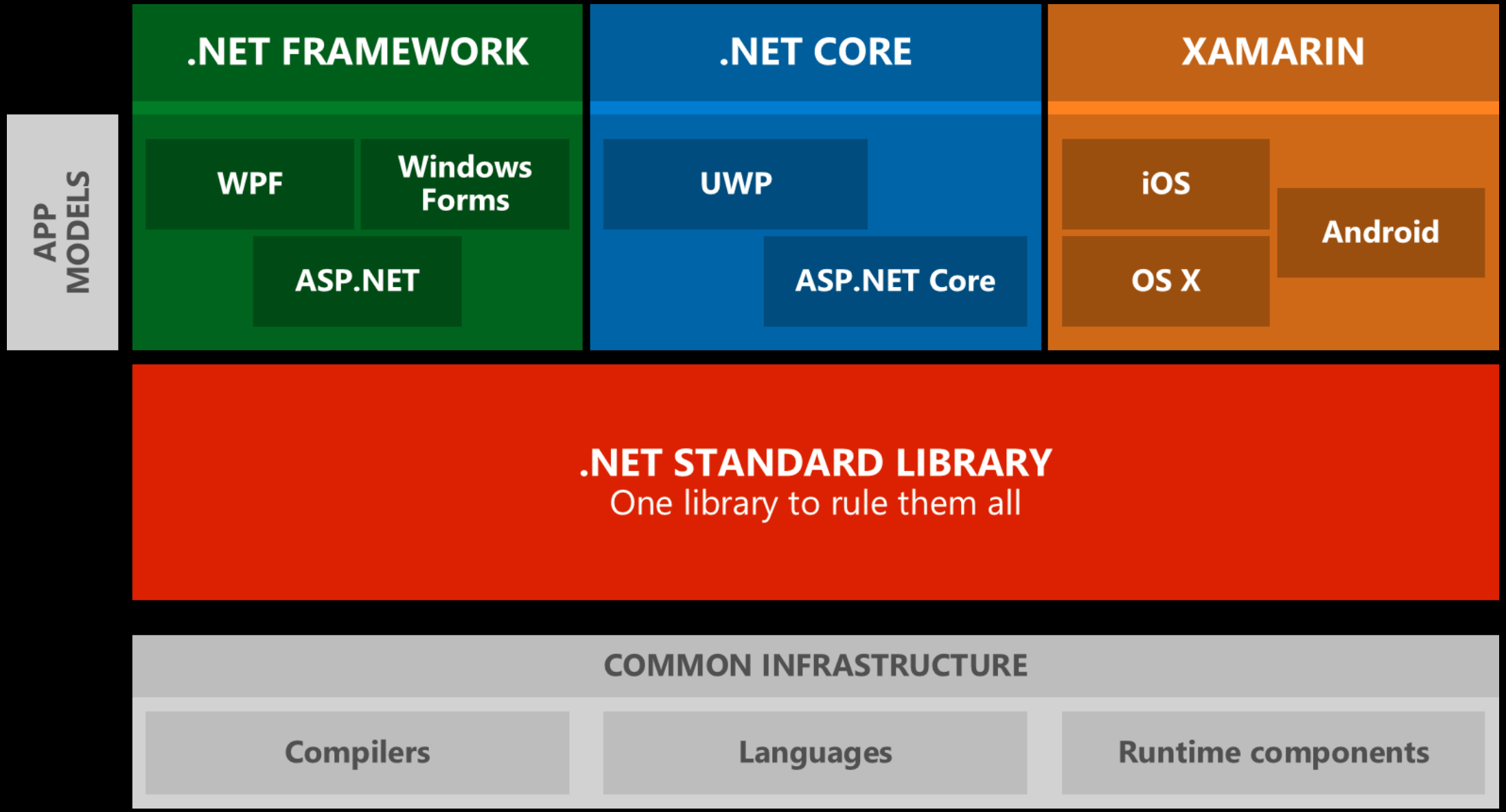
@dylanbeattie

- Building websites since 1992
- Systems architect at www.spotlight.com
- Website: www.dylanbeattie.net
- dylan@dylanbeattie.net

14:00	TALK: Intro & Background
14:30	CODE: building the “Hello, World” API
	(break)
15:00	TALK: Scaling Security
15:20	CODE: Implementing API security
	(break)
15:40	TALK: Logging
16:00	CODE: Implement logging in your API
	(break)
16:20	TALK: Discoverability, Endpoints and Monitoring
16:40	CODE: Implementing discovery and status endpoints
	(break)
17:00	TALK: Deployment and Configuration
17:30	TALK: Going Serverless

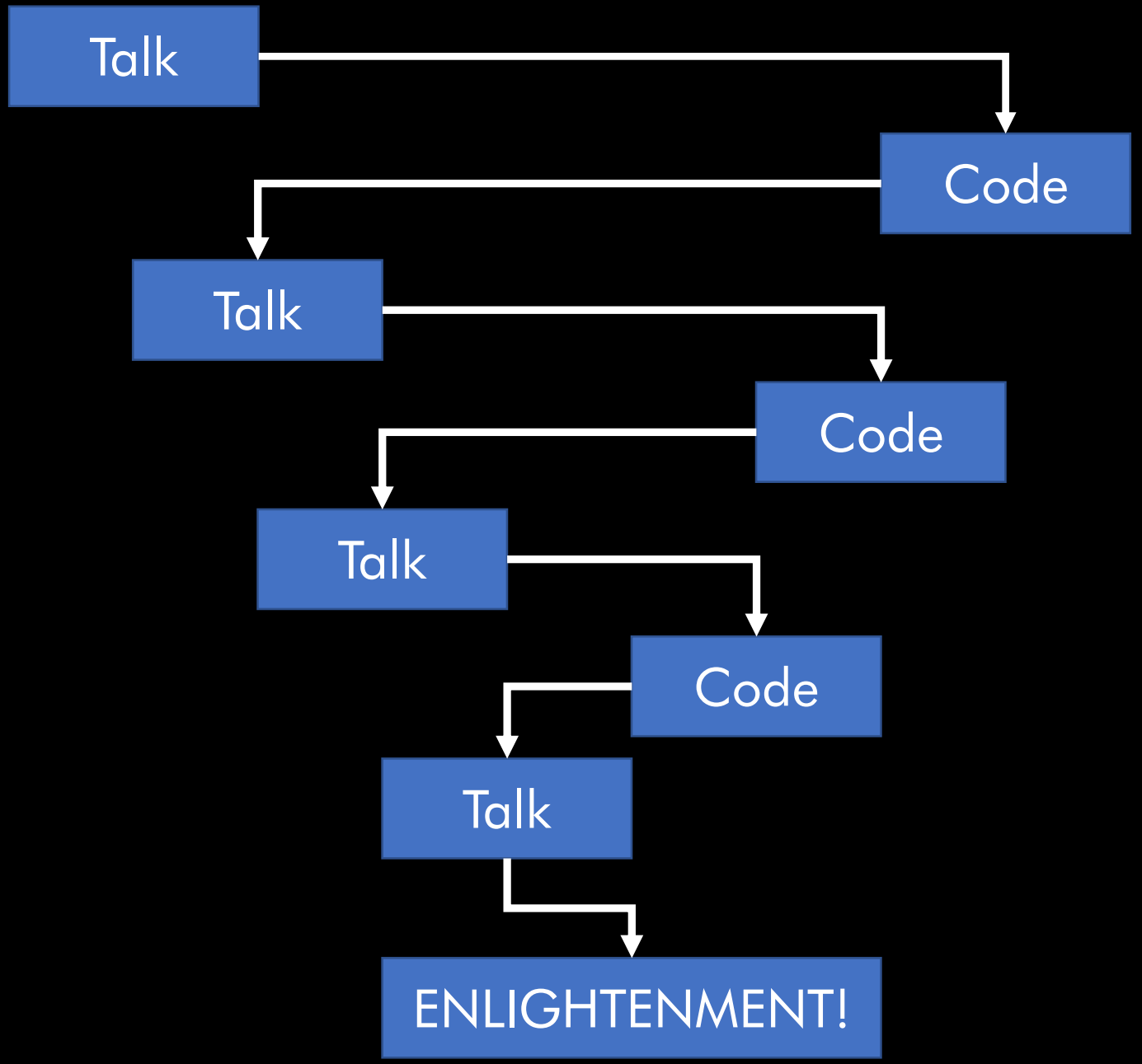
Intro & Background

Systems, Scaling, and the State of .NET in 2017


APP MODELS
.NET FRAMEWORK
WPF
Windows Forms
ASP.NET
.NET CORE
UWP
ASP.NET Core
XAMARIN
iOS
Android
OS X
.NET STANDARD LIBRARY
One library to rule them all
COMMON INFRASTRUCTURE
Compilers
Languages
Runtime components


works on
my
machine
beatsaudio
Pavilion

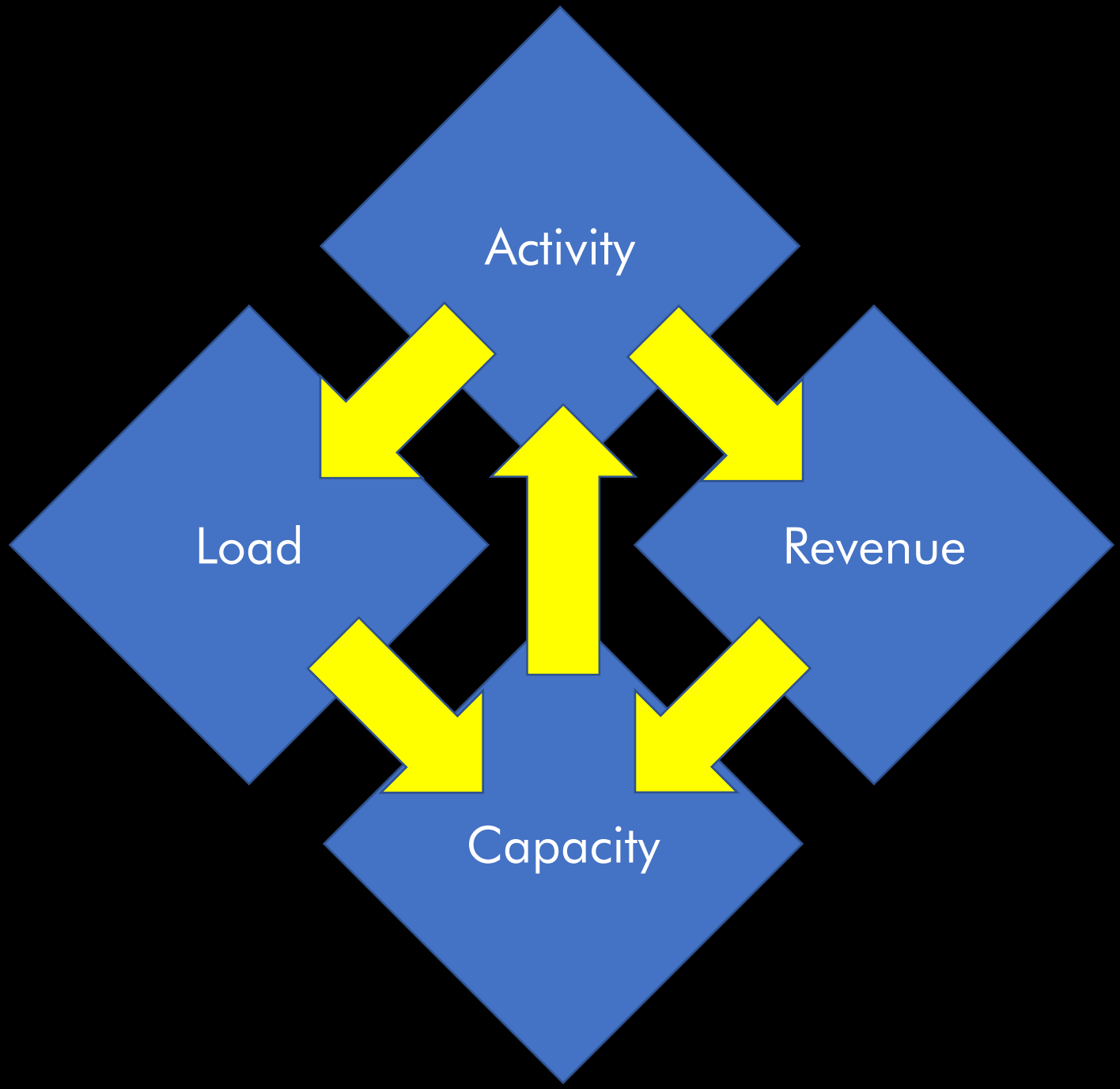

works on my machine
works on my machine
works on my machine


Talk
Code
Talk
Code
Talk
Code
Talk
ENLIGHTENMENT!


DELL
PowerEdge R610
SAS
300GB 10k


DELL
PowerEdge 2970
SAS 36GB 10K
SAS 36GB 10K
SAS 36GB 10K
SAS 36GB 10K
0
1
2
3
4
5
6
7


PowerEdge T320
PowerEdge R520
PowerEdge R520
PowerEdge M1000e


Activity
Load
Revenue
Capacity

(a brief tangent
about databases)

```
CREATE TABLE Thing (
    ThingId INT IDENTITY(1,1) PRIMARY KEY,
    ThingName varchar(64),
    CreatedAt datetimeoffset
)
```

Hang on… we might end up with more than 2 billion Things.

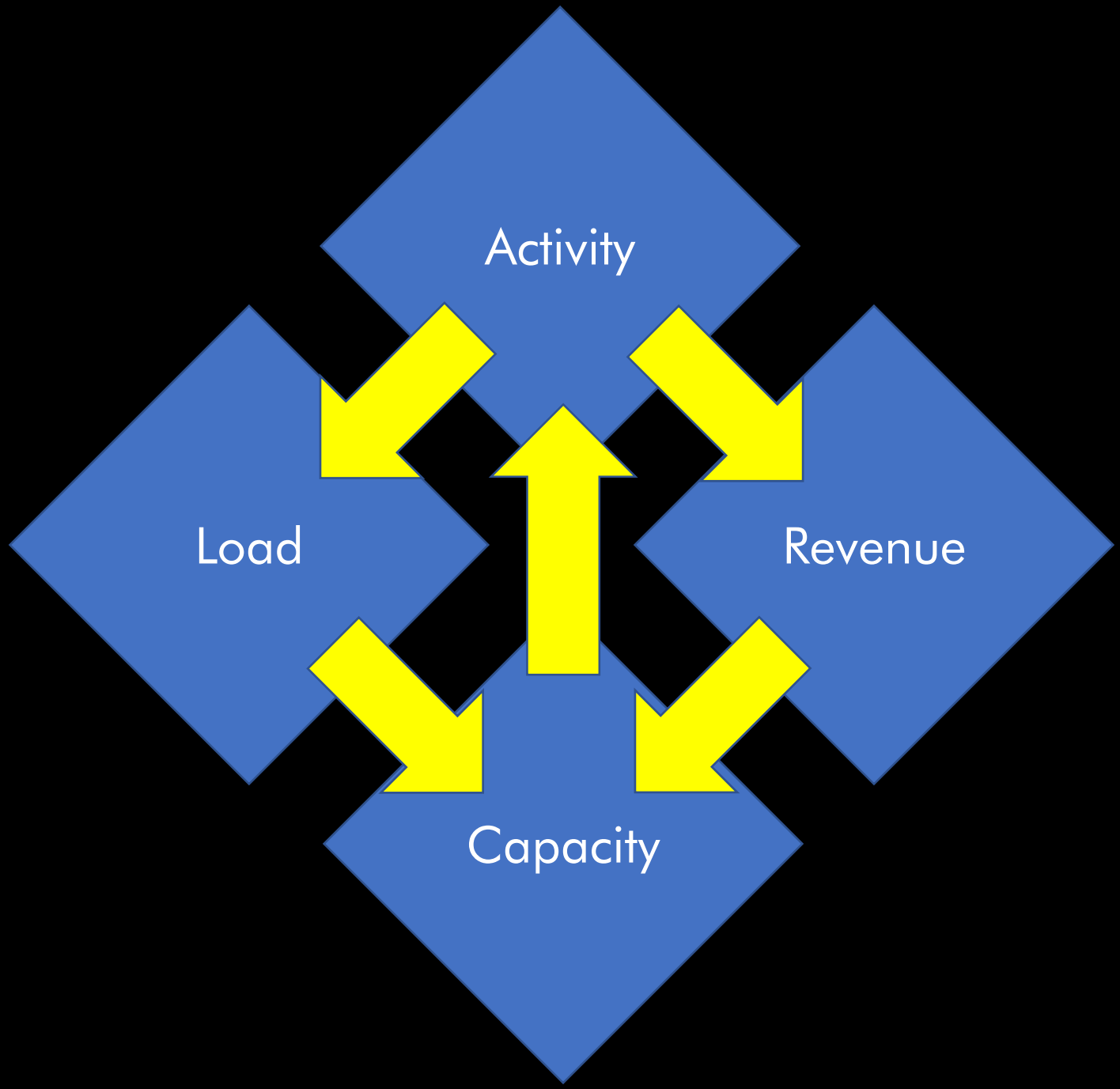
```
CREATE TABLE Thing (
    ThingId BIGINT IDENTITY(1,1) PRIMARY KEY,
    ThingName varchar(64),
    CreatedAt datetimeoffset
)
```

```
CREATE TABLE Thing (
    ThingId BIGINT IDENTITY(2147483648,1)
        PRIMARY KEY,
    ThingName varchar(64),
    CreatedAt datetimeoffset
)
```

Existing requirement: X

Future requirement: Y

How can we make Y happen now?


Activity
Load
Revenue
Capacity

Exercise 1

Implement the “Hello, World” API

GET	/greetings	Return a list of all the greetings available in the system
POST	/greetings	Add a new greeting to our system.
GET	/greetings/{id}	Return a “Hello, World!” style greeting in the specified format
GET	/greetings/{id}?name={name}	Return a “Hello, {name}” style greeting
PUT	/greetings/{id}	Replace the greeting at {id} with the supplied greeting
DELETE	/greetings/{id}	Delete the specified greeting

Scaling Security

As: a user,

I want to: enter my username and password

So that: I can access secure areas of the application

As: a user,

I want to: ~~enter my username and password~~

~~So that:~~ ~~I can~~ access secure areas of the application


007
DIE ANOTHER DAY
11.22

Security in three rules

1. Keep MY STUFF safe
2. Let ME see MY STUFF
3. Don't share MY STUFF with anybody else *(unless I say it's OK)*

Security, services and scalability: Models

1. No security. Completely wide open.
2. Manual security. Limited number of clients, managed manually
3. Application security. Users are employees / developers / partners
4. Consumer security. Users are customers.
5. Federated security. Users are somebody else's customers.

Security, services and scalability: Mechanisms

1. Obfuscated endpoints
2. Credentials
    1. Per-request, e.g. HTTP Basic
    2. Session-based
3. Token-based, e.g. OAuth2
4. Certificate-based
5. One-time pad / shared secret / out-of-band authentication

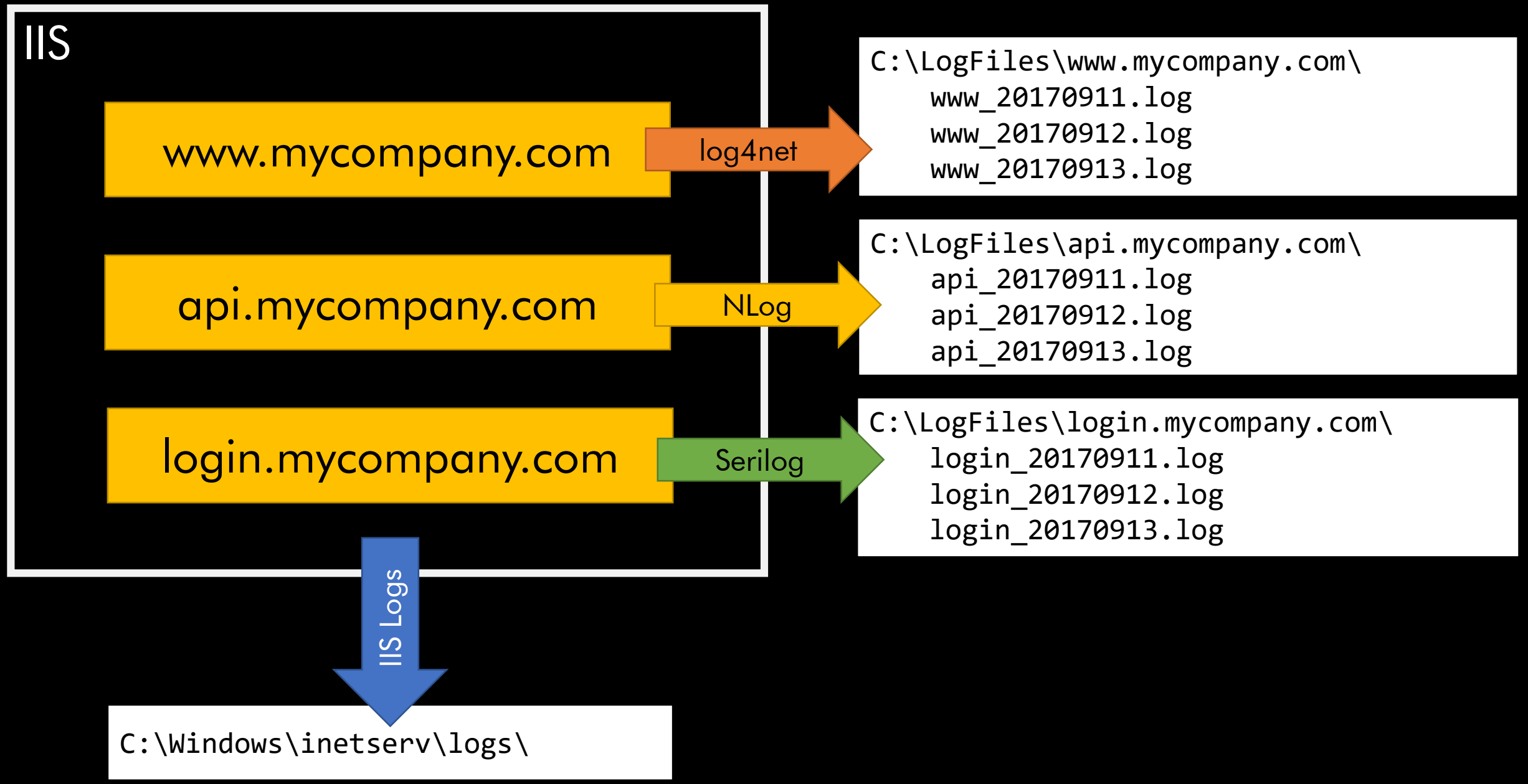
Scaling Security : Questions

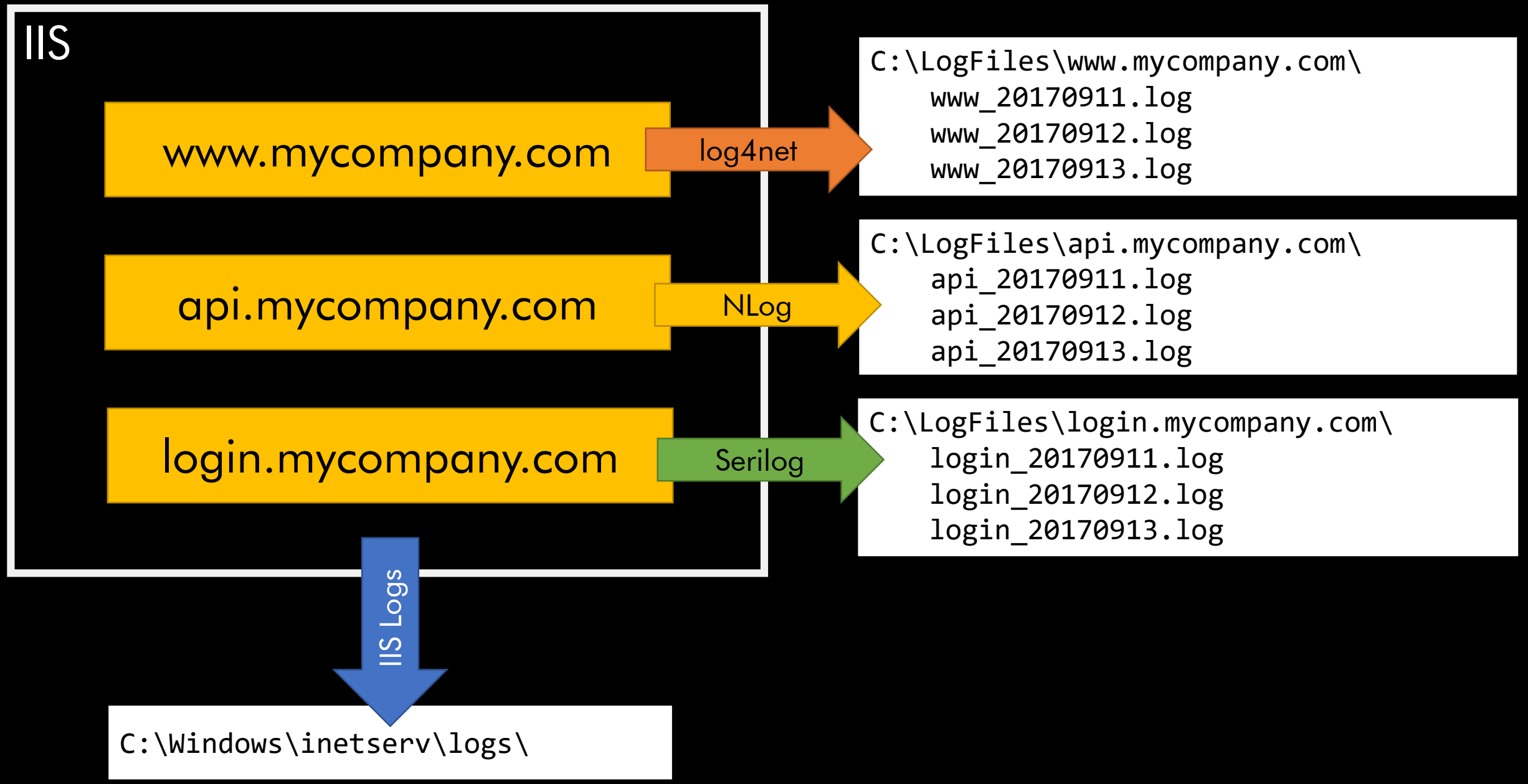
- What's the worst that can happen?
- Look for asymmetric scenarios
  - E.g. revoking credentials vs reinstating
- What can you do WITHOUT security?
  - API endpoints?
  - Personalization?
  - Donut caching?

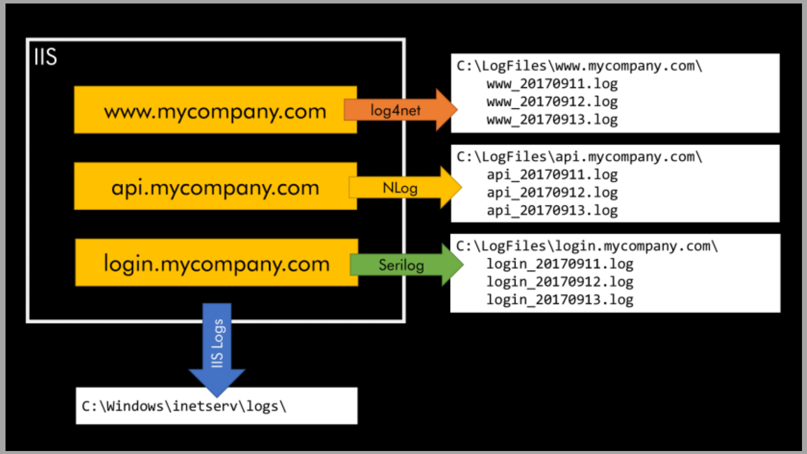
Exercise 2

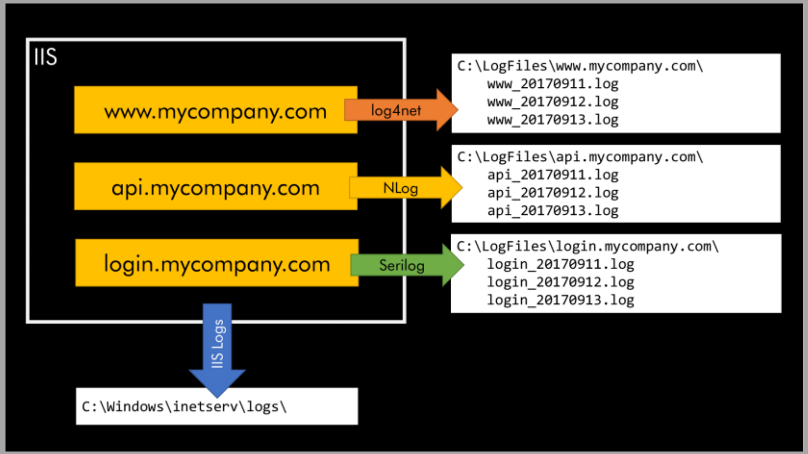
Securing the “Hello, World” API

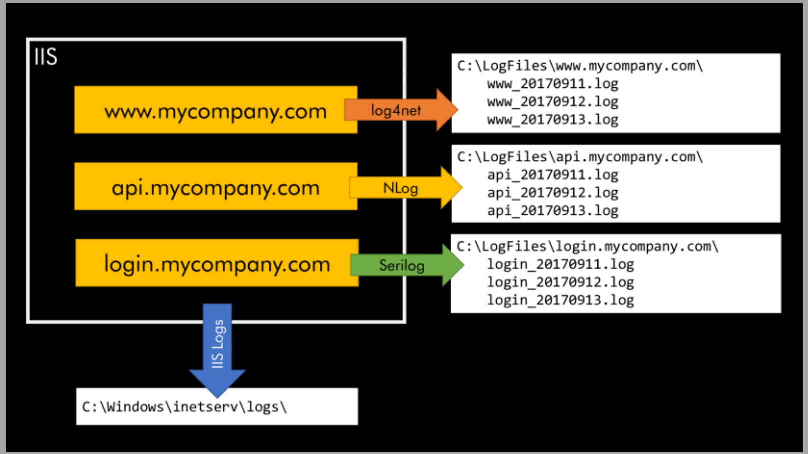
Logging: How?

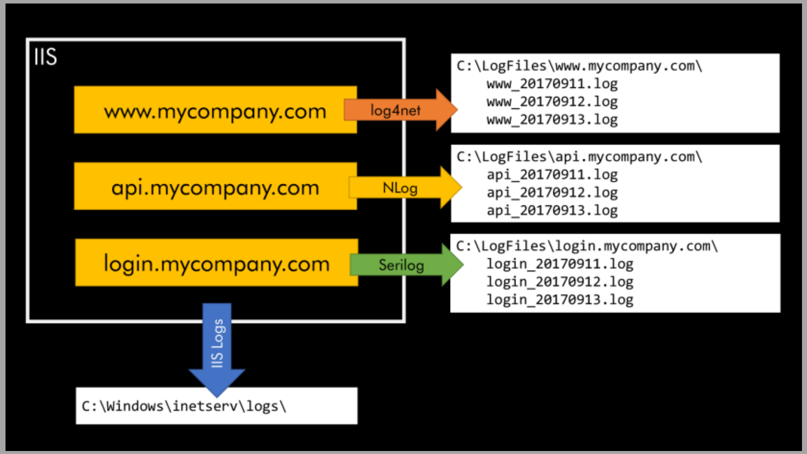

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

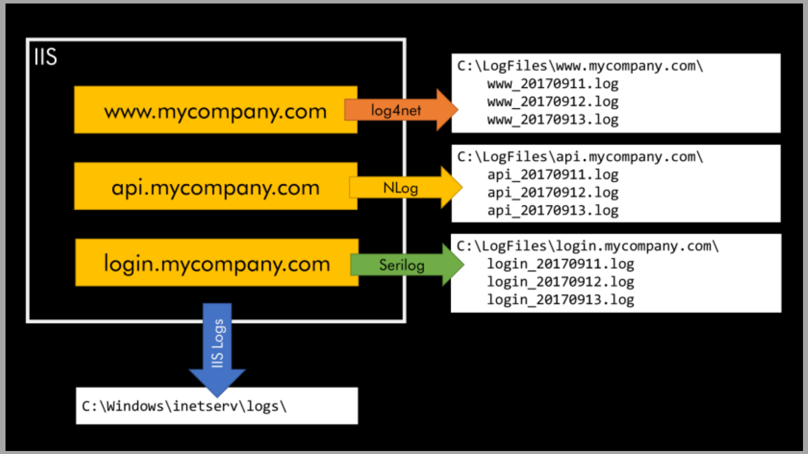

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

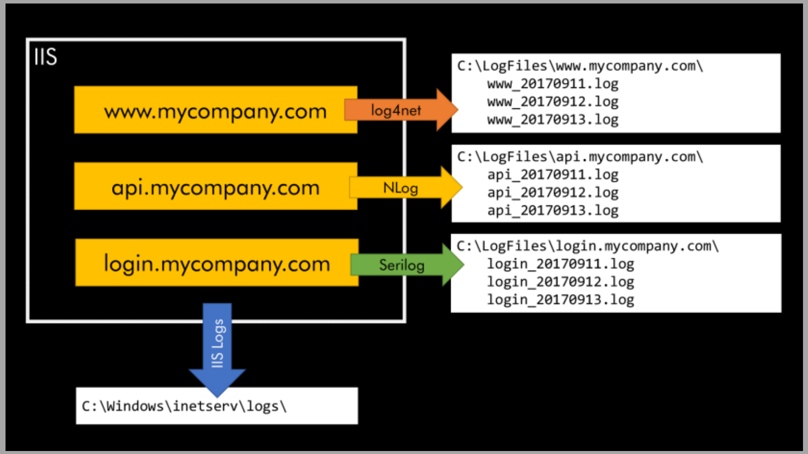

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

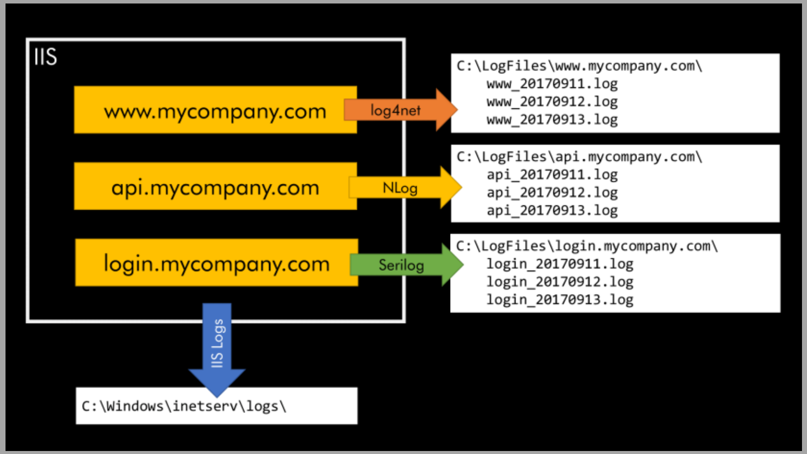

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

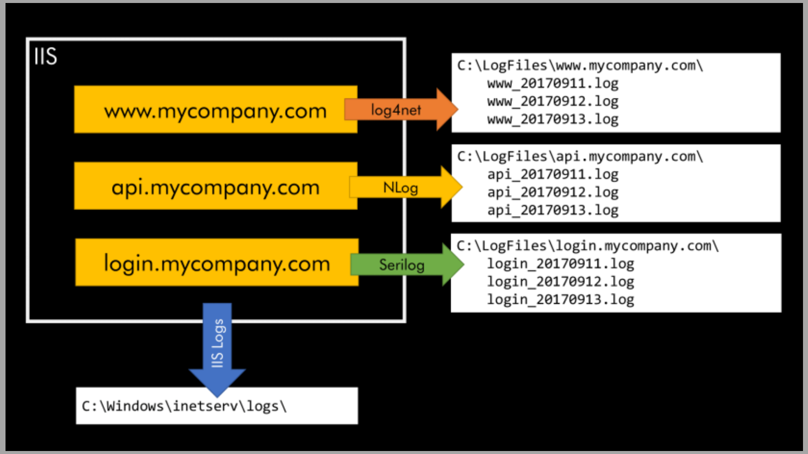

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

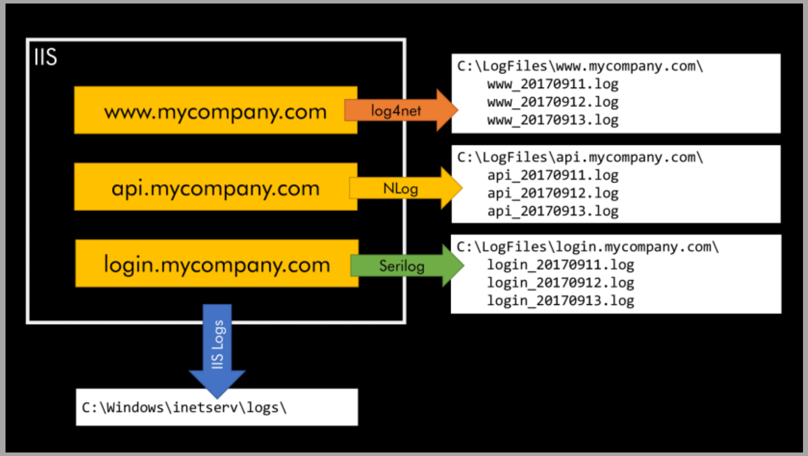

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

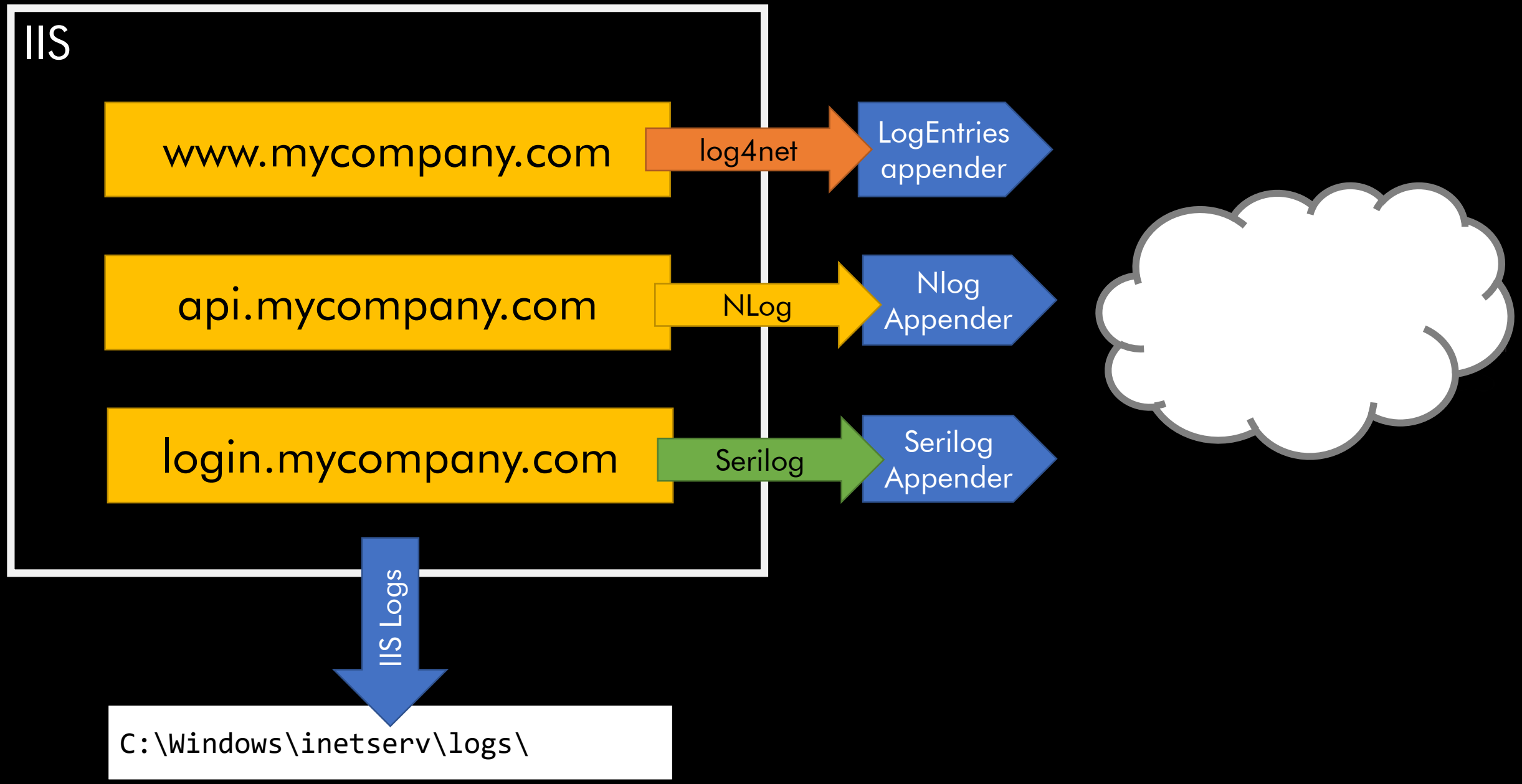

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

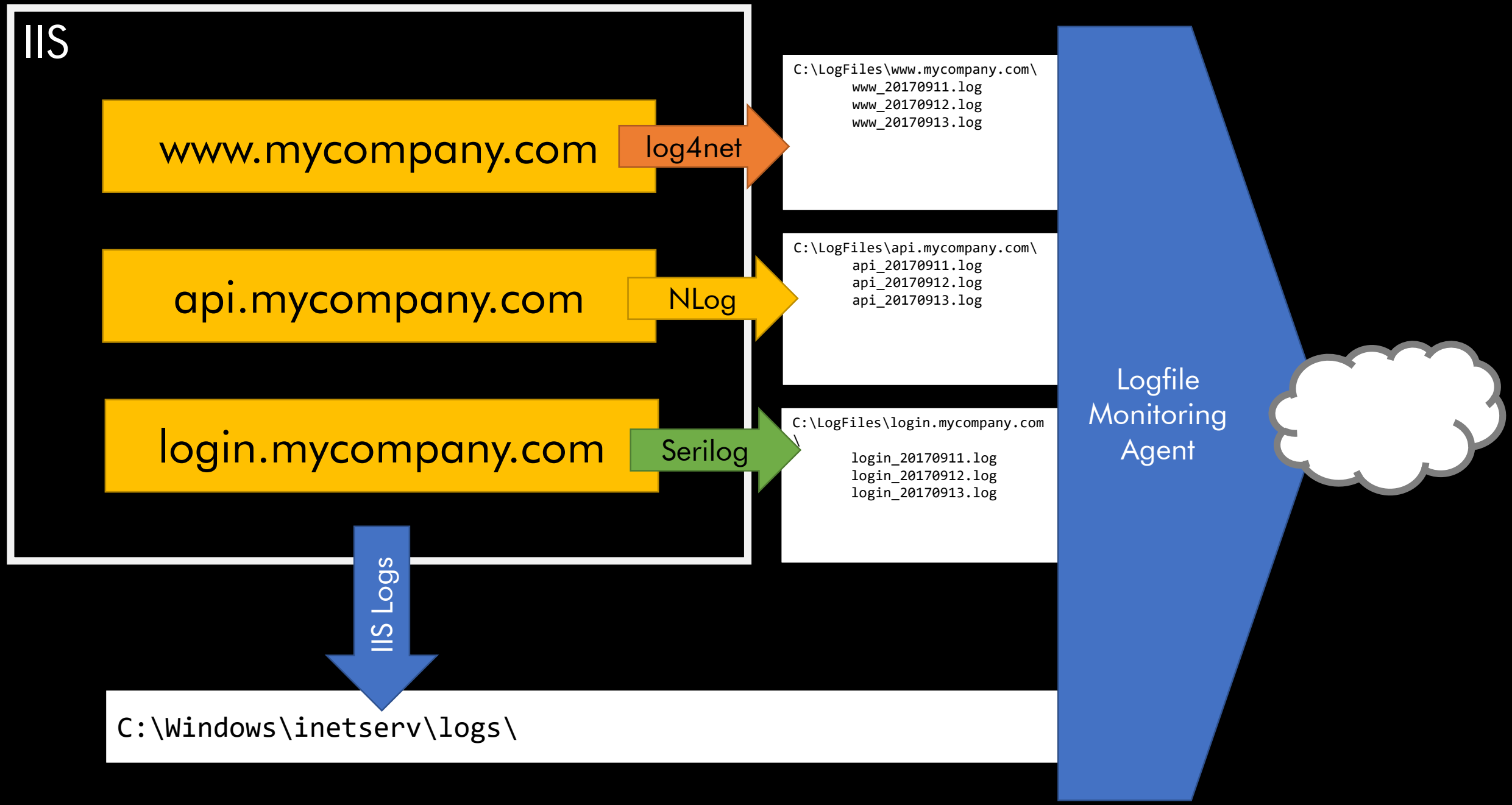

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

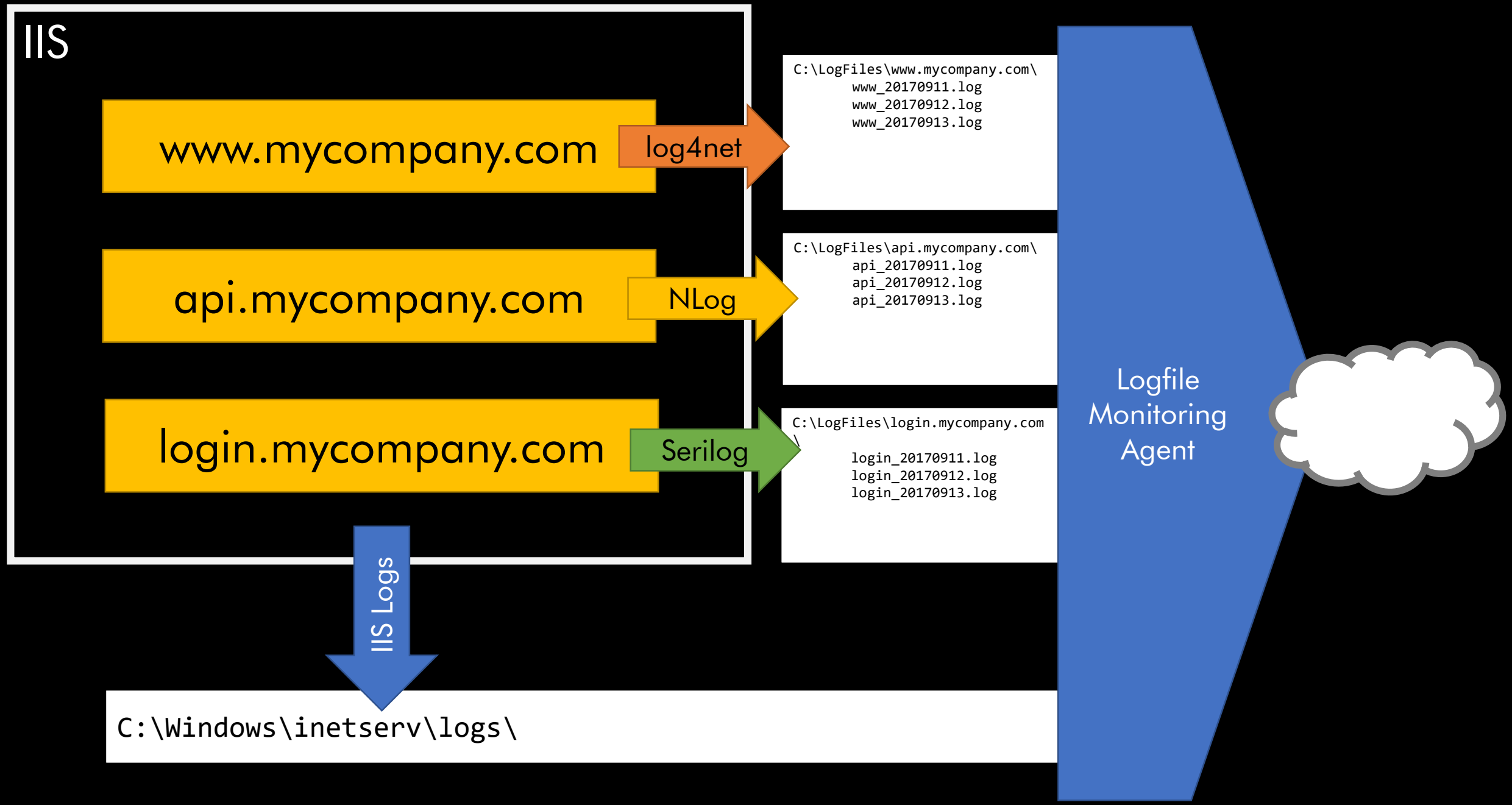

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

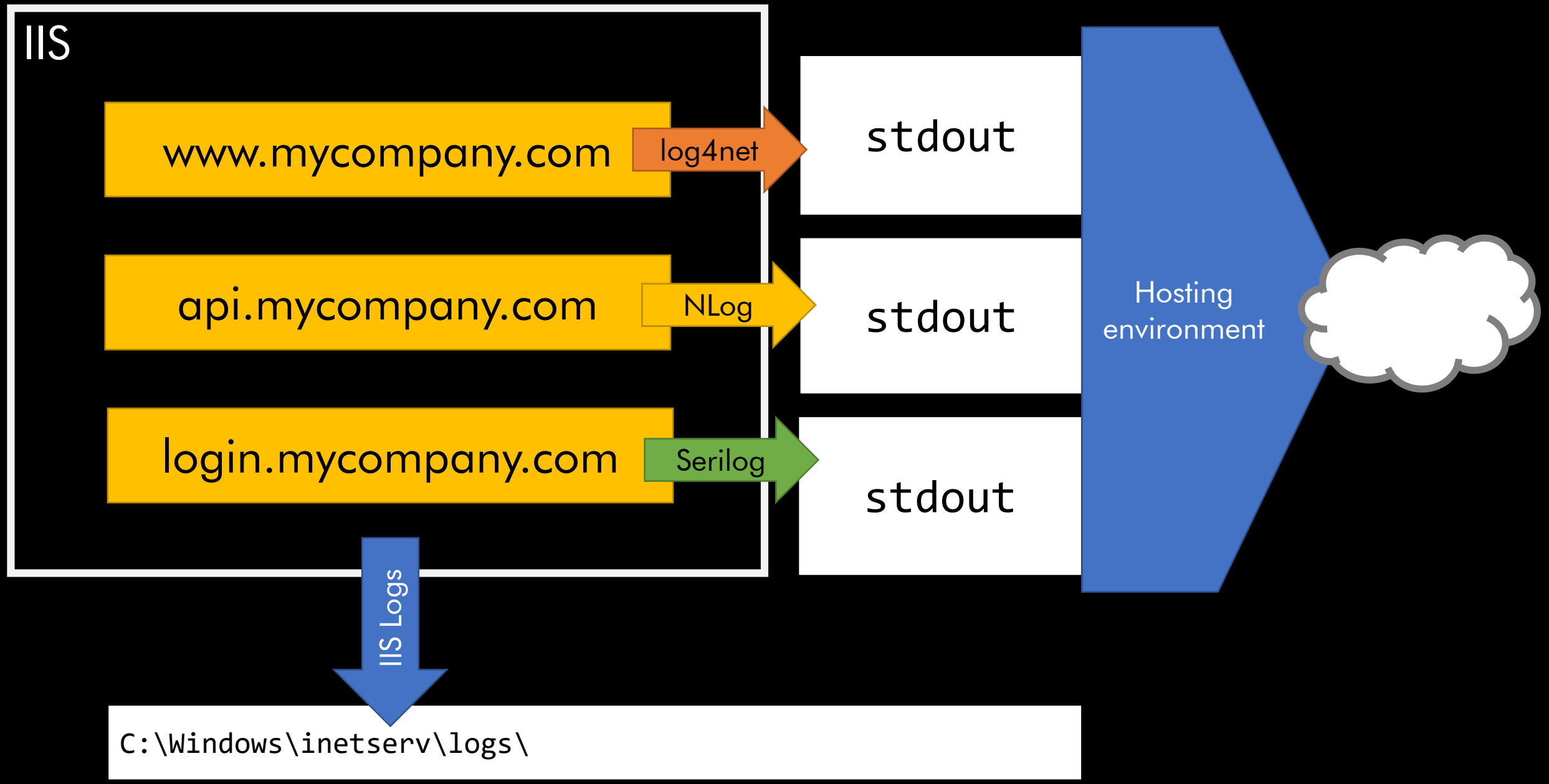

IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\


IIS
www.mycompany.com
log4net
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
api.mycompany.com
NLog
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
login.mycompany.com
Serilog
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
IIS Logs
C:\Windows\inetserv\logs\

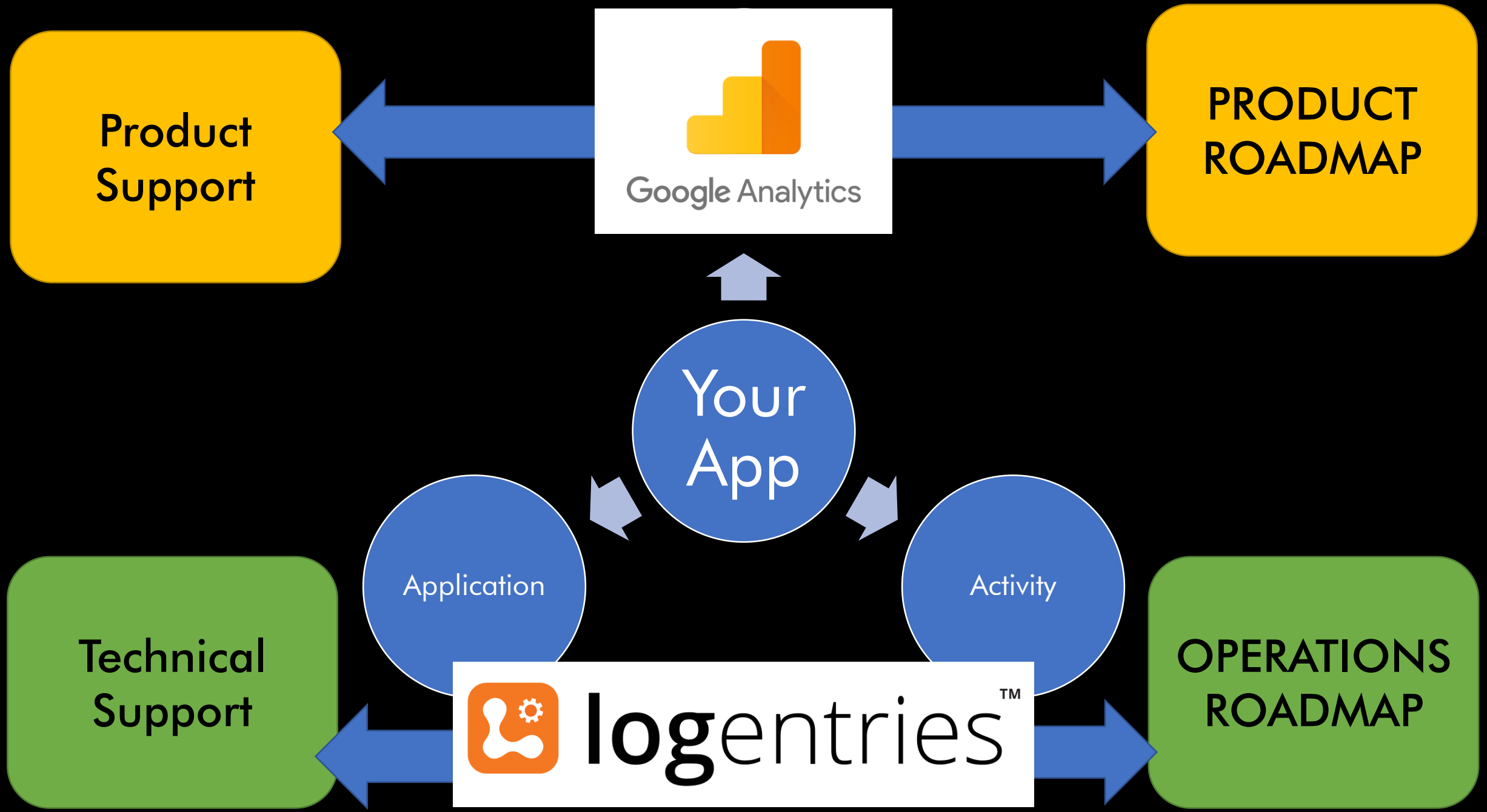

IIS
www.mycompany.com
log4net
LogEntries appender
api.mycompany.com
NLog
Nlog Appender
login.mycompany.com
Serilog
Serilog Appender
IIS Logs
C:\Windows\inetserv\logs\

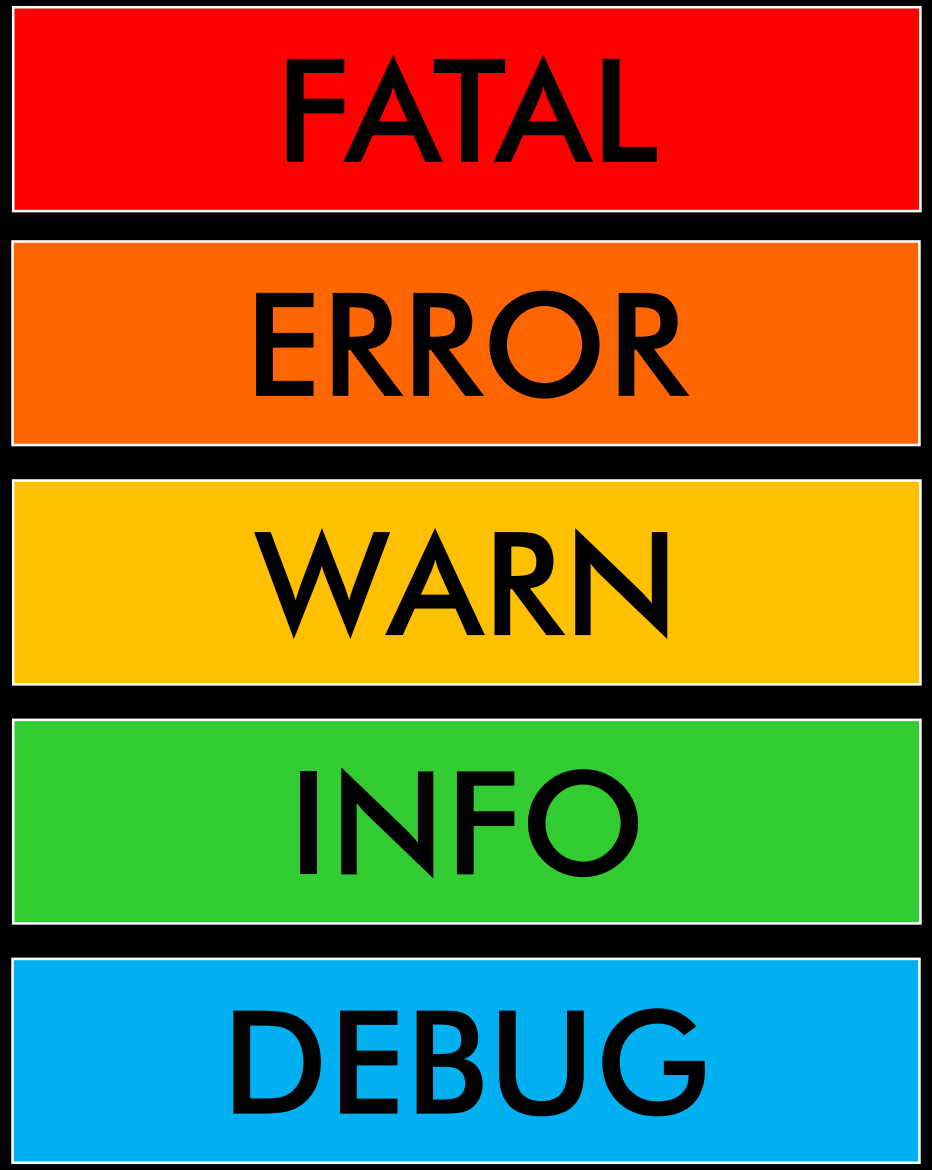

IIS
www.mycompany.com
log4net
api.mycompany.com
NLog
login.mycompany.com
Serilog
IIS Logs
C:\Windows\inetserv\logs\
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
Logfile Monitoring Agent


IIS
www.mycompany.com
log4net
api.mycompany.com
NLog
login.mycompany.com
Serilog
IIS Logs
C:\Windows\inetserv\logs\
C:\LogFiles\www.mycompany.com\
www_20170911.log
www_20170912.log
www_20170913.log
C:\LogFiles\api.mycompany.com\
api_20170911.log
api_20170912.log
api_20170913.log
C:\LogFiles\login.mycompany.com\
login_20170911.log
login_20170912.log
login_20170913.log
Logfile Monitoring Agent


IIS
www.mycompany.com
log4net
api.mycompany.com
NLog
login.mycompany.com
Serilog
stdout
stdout
stdout
Hosting environment
IIS Logs
C:\Windows\inetserv\logs\

Logging: What?


Product Support
Google Analytics
PRODUCT ROADMAP
Your App
Application
Activity
Technical Support
logentries™
OPERATIONS ROADMAP


FATAL
ERROR
WARN
INFO
DEBUG

FATAL

- Application is completely unresponsive
- Multiple users affected with immediate effect
- Immediate attention. Stop what you're doing and look into it.

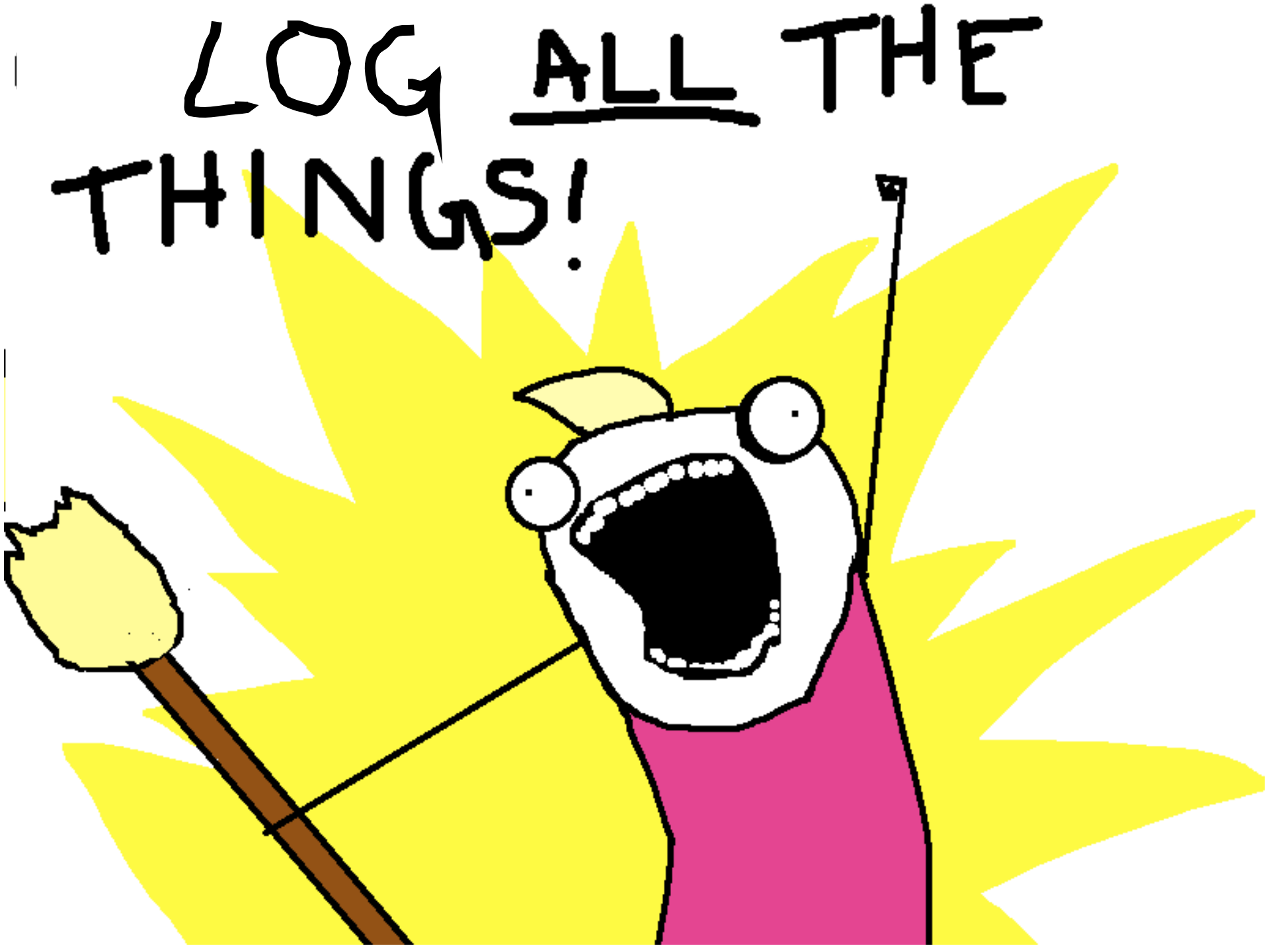
ERROR & WARN

- They *will* happen.
- **ERROR** – one person noticed, maybe?
- **WARN** – nobody noticed but it's still odd.
- Individual messages probably aren't useful

https://universitycompare.com/articles/worst-commuters-sit-next-tube/

INFO

- Everything's fine
- Startup, shutdown, cache recycle
- In a perfect world, INFO is all you'll ever see.



http://hyperboleandahalf.blogspot.co.uk/

wait... all the things?

http://hyperboleandahalf.blogspot.co.uk/


3:17 AM

```
log.Fatal Wake_Me_Up_At_4AM_On_A_Sunday()

Log.Error Apologize_To_User_And_Raise_A_Ticket()

log.Warn  Make_A_Note_In_Case_It_Happens_Again()

log.Info  Everything_Is_Fine_Just_Checking_In()

log.Debug Fill_My_C_Drive_With_Stack_Traces()
```

Adapted from a post by Daniel Lebrero / https://labs.ig.com/logging-level-wrong-abstraction

Exercise 3

Centralised Logging for Hello, World.

Discoverability

Exercise 4

Discoverability and Monitoring

Deployment and Configuration

The Twelve-Factor App

12FA #1: Codebase

- One app = one codebase
- Exploit natural domain / process boundaries
- Use revision control

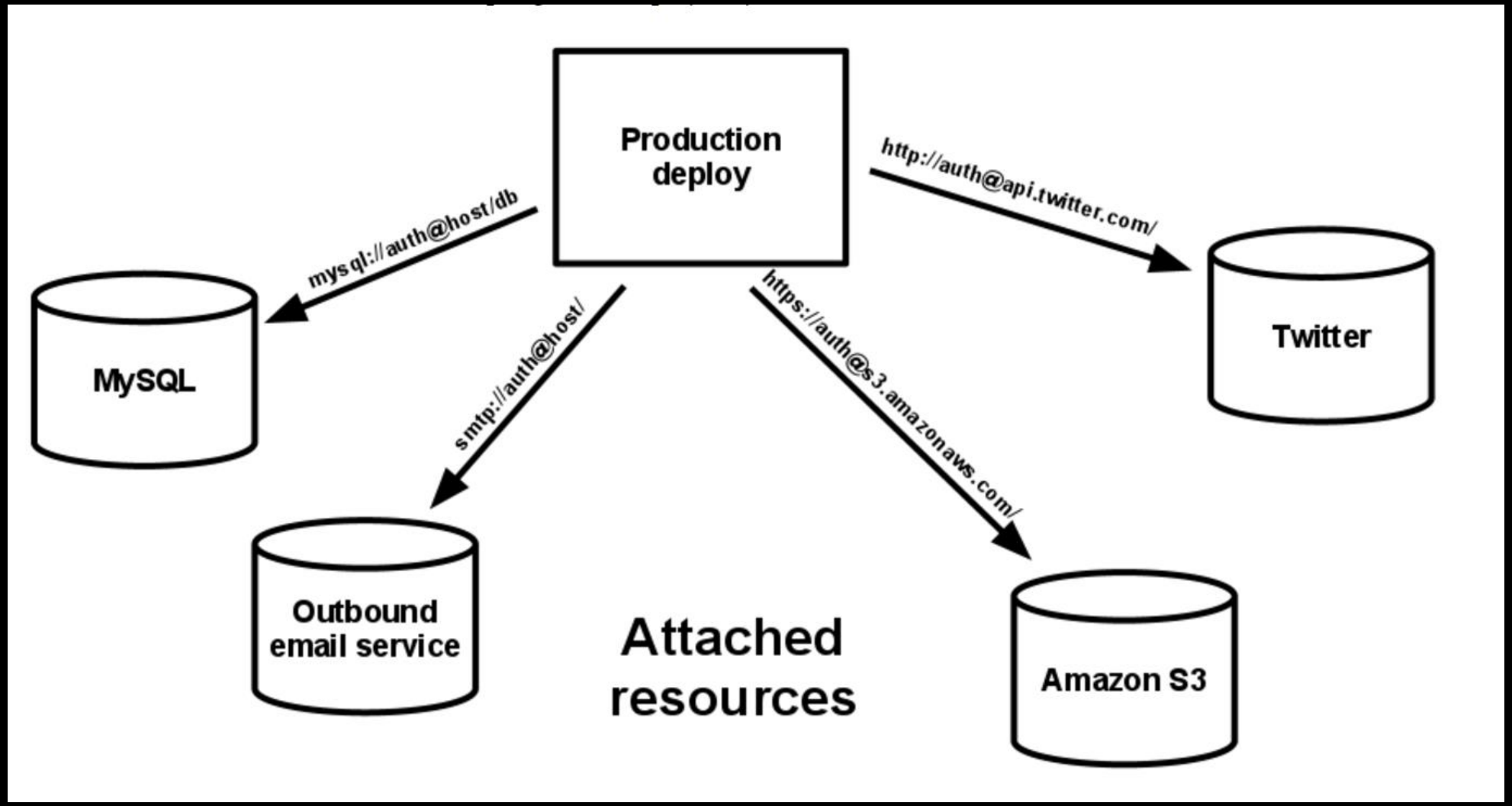
12FA #2: Dependencies

- Explicit
- Declared
- Isolated
- Code dependencies – packaged with your codebase
  - NuGet, Paket, npm
  - Don't assume ANYTHING
    - Except the stuff that's part of the environment
      - Which is the stuff you can assume
        - Because you manage the environment
          - Except the bits you don't
            - Look, dependencies are complicated, OK? :/

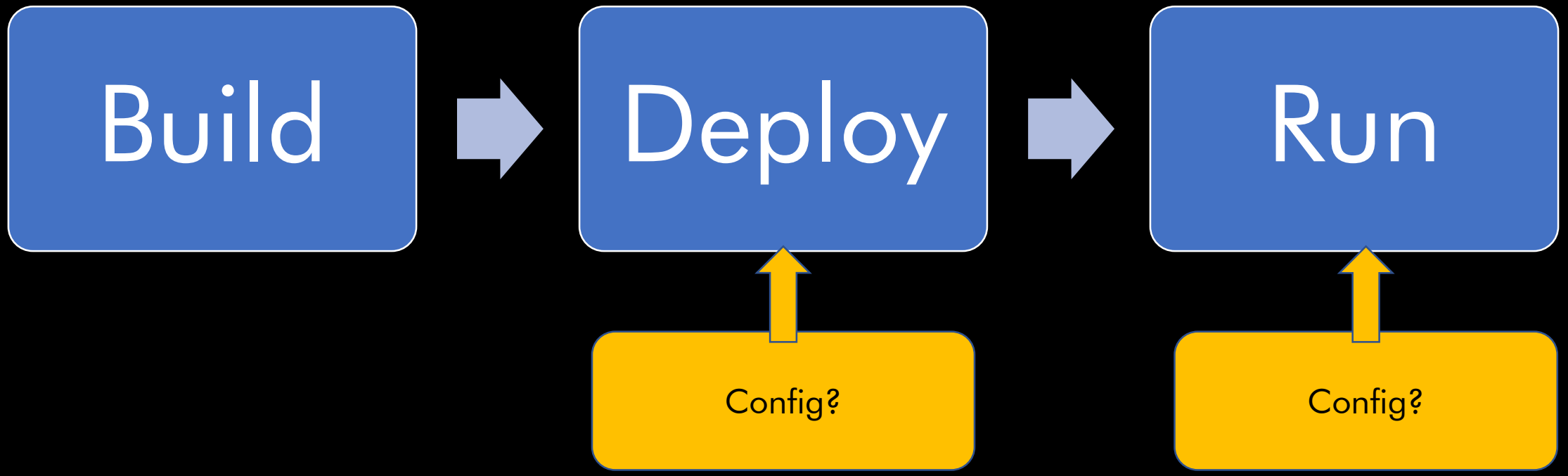
12FA #3: Configuration

1. Constants; hard-coded values; web.config
2. Multiple configurations
    1. Web.Debug.config vs Web.Release.config
    2. Config held in source control
    3. Build pipeline produces configured packages
3. Configuration via deployment
    1. E.g. Octopus Deploy XML transforms
4. Configuration via environment (machine level)
5. Configuration via environment (network level)

12FA #4: Backing Services



21FA #5: Build, Release, Run



On Builds, Releases and Versioning

- Every release has a unique release number
- Release numbers increase over time
  - Lower numbers are ALWAYS earlier releases
- Incrementing build numbers – 1,2,3,etc
- Datetime stamps – 201709131410, 201709131425
- Semantic versioning

12FA #6: Stateless Processes

- Any persistence has to be in an isolated service
  - Database / data store
    - Memcached
    - Redis
  - Object storage (filesystem share, S3 bucket, cloud)
  - Don't rely on ANYTHING from a previous operation still existing in a future operation
    - No temp files
    - No local storage
    - No sessions
    - No in-memory

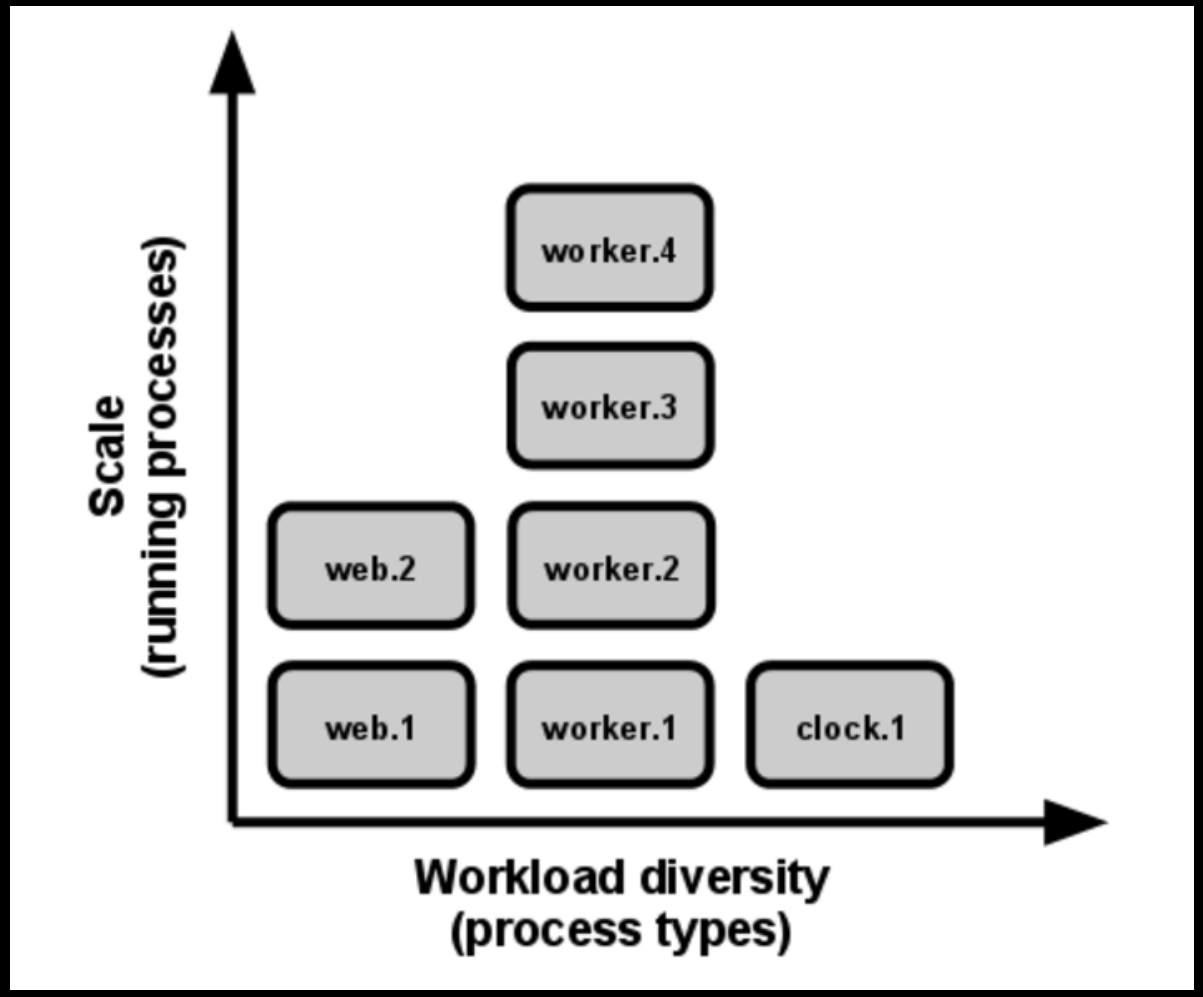
12FA #7: Port Binding

.NET App — Kestrel

Python app — Tornado

Java app — Jetty

12FA #8: Concurrency



12FA #9: Disposability

- Start up FAST
- Shut down GRACEFULLY
  - Stop listening for new requests
  - Finish handling any existing requests
- Consider a crash-only approach

12FA #10: Dev/Prod Parity

- The time gap:
  - A developer may work on code that takes days, weeks, or even months to go into production.
- The personnel gap:
  - Developers write code, ops engineers deploy it.
- The tools gap:
  - Developers may be using a stack like Nginx, SQLite, and OS X, while the production deploy uses Apache, MySQL, and Linux.

12FA #11: Logs are Event Streams

- Log everything to STDOUT
- Let the environment worry about monitoring, aggregating, searching, etc

12FA #12: Admin processes

- Ship your ad-hoc admin scripts
- Make them releases
- Run them using the same environment and configuration as real code

Exercise 5

Going Serverless

Hello, Galaxy!

Hello World at Scale

Dylan Beattie

Progressive.NET Tutorials 2017

@dylanbeattie #prognet