Building Networking At Flipkart

Siddharth Gupta
@sids7

Work Philosophy

- We are very flexible in terms of processes.
- Things that work for one team might not necessarily work for the other.
- This same thing goes for languages, frameworks etc.
- People are free to choose these if it best fits the cause and it maintainable by others in the team.

Tools We Use

GitHub

Atlassian JIRA Agile

fastlane

crashlytics

JIRA

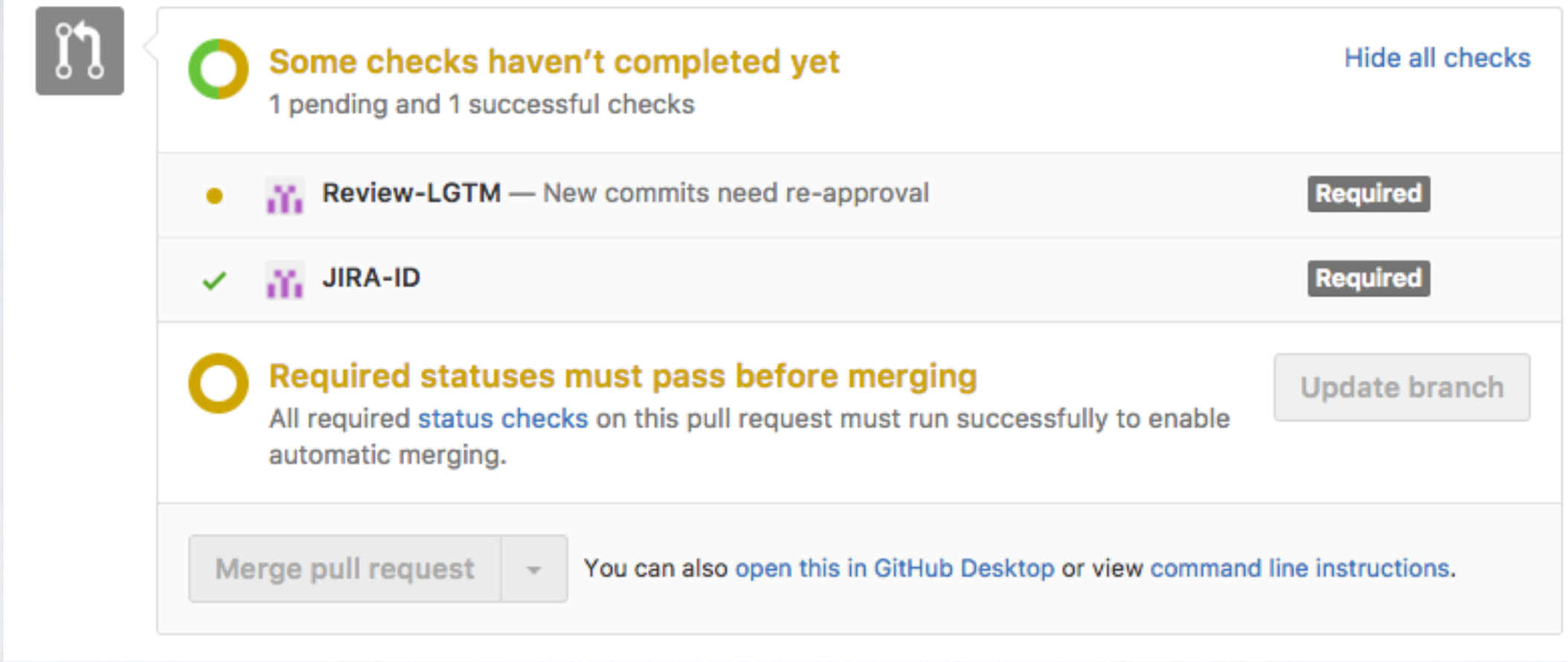
Scrum Board

7 days remaining Complete Sprint

QUICK FILTERS: Product UI Server Only My Issues Recently Updated

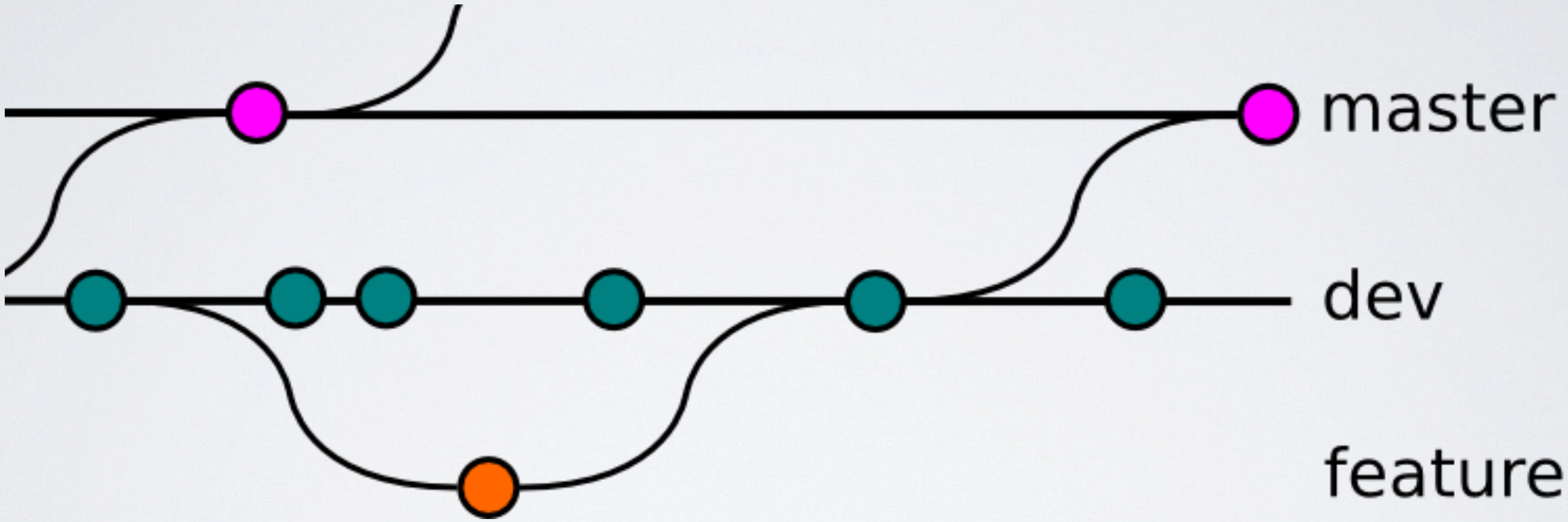
2 To Do	3 In Progress	2 Code Review	1 Done
TIS-68 Homepage footer uses an inline style - should use a class — Large Team Support 4	TIS-49 Draft network plan for Mars Office — Local Mars Office 5	TIS-69 Add a string anonymizer to TextUtils — Large Team Support 1	TIS-8 Requesting available flights is now taking > 5 seconds — Large Team Support 2
TIS-16 Establish relationship with local office supplies company — Local Mars Office 9	TIS-17 Engage Saturn's Rings Resort as a preferred provider — Space Travel Partners 7	TIS-67 Developer Toolbox does not display by default — Large Team Support 5	TIS-66 Add pointer to main css file to instruct user to create child themes — Large Team Support 4
	TIS-30 Create Saturn Summer Sizzle Logo — Summer Saturn Sale 1		TIS-45 Email non registered users to sign up with Teams in Space — Summer Saturn Sale 8
	TIS-23 Engage JetShuttle SpaceWays for short distance space travel — Local Mars Office 1		

GitHub



LGTM: Looks Good To Merge

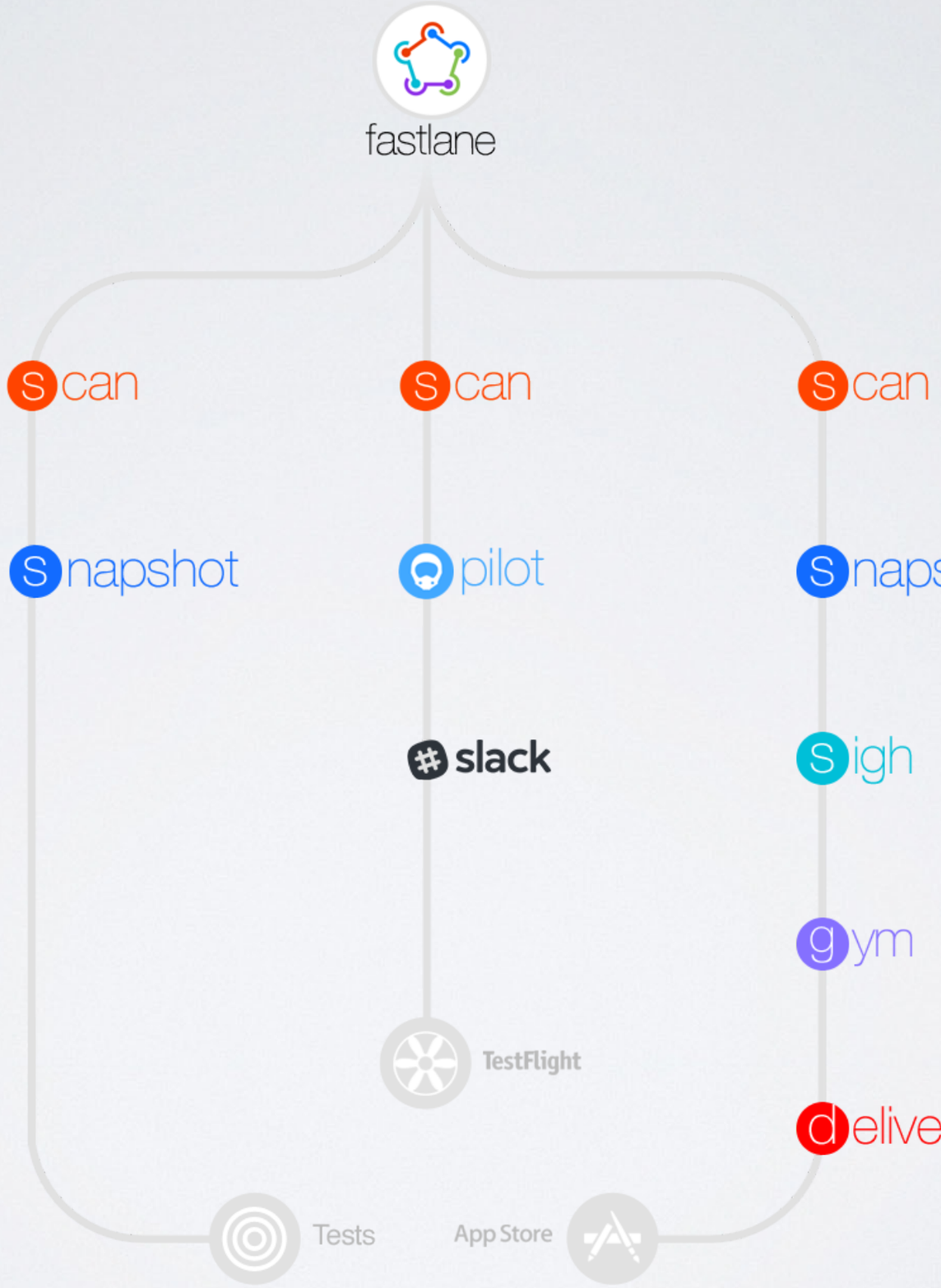
Git Branching Strategy



master and **dev** are protected branches

No direct commits into master/dev, master is sacrosanct, dev should never be broken

Fastlane


fastlane
scan
scan
scan
snapshot
pilot
snapshot
slack
sigh
gym
TestFlight
deliver
Tests
App Store

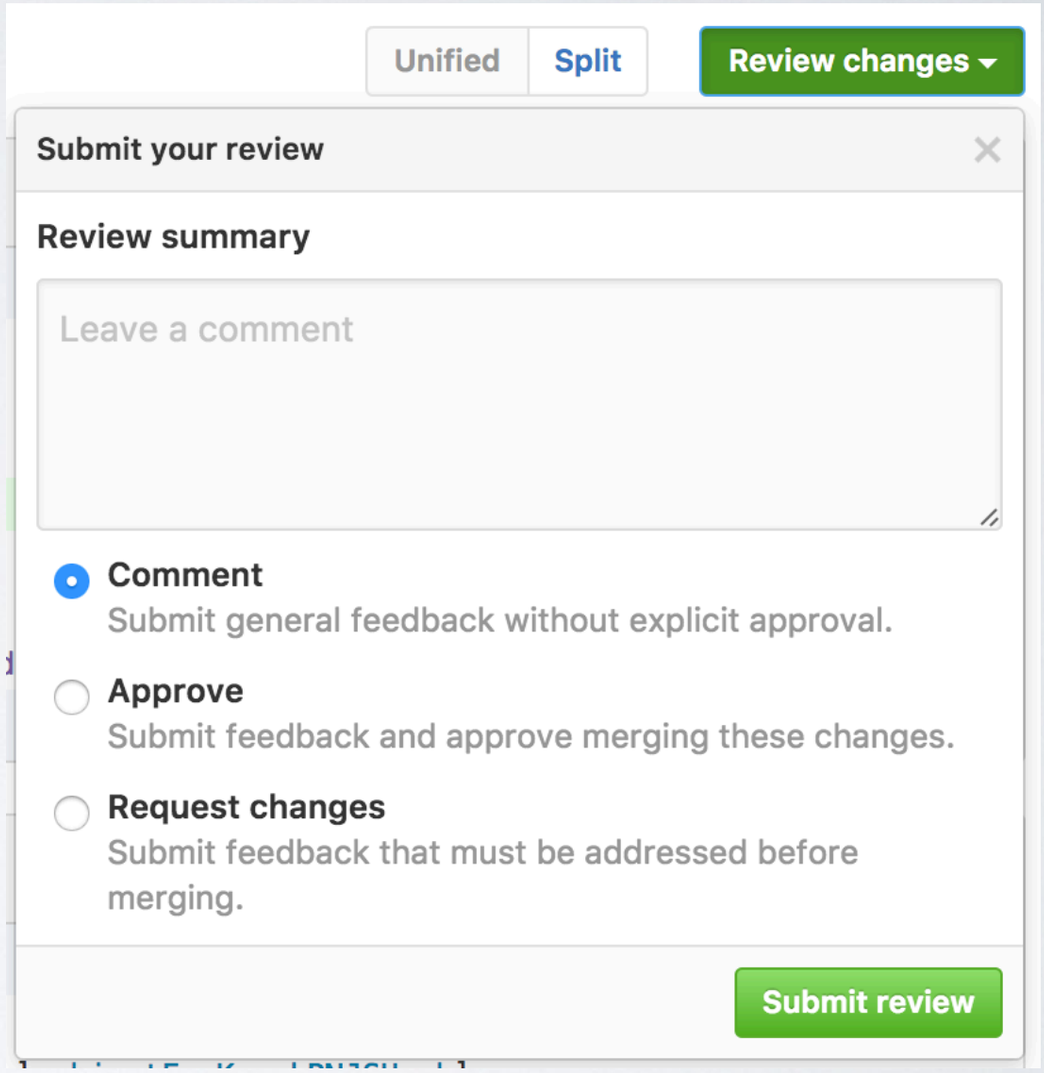
Dependency Management


```
use_frameworks!
```

+ Private Cocoapod Specs Repo for Internal Pods

Code Review

- Strict Code Reviews
- Functional + Design Reviews
- Shadowed Code Reviews
- LGTM Hygiene



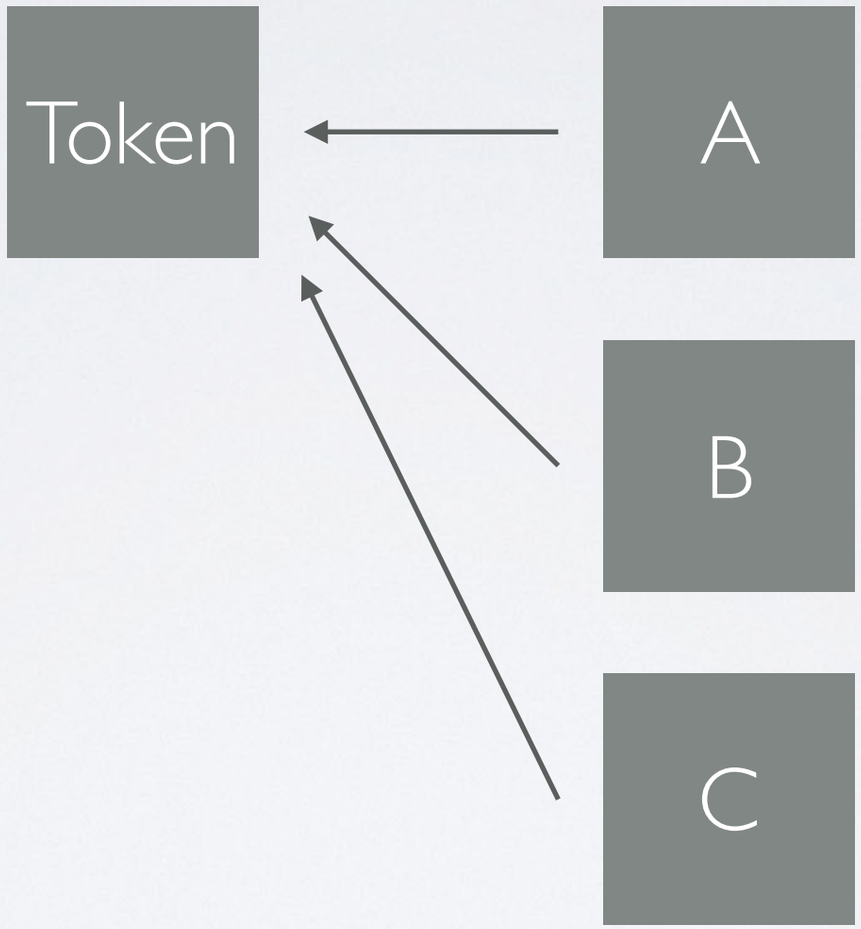
Release to AppStore

- Deliberately Manual
- Release Checklist
- A Lot of Dependencies on Other Teams + Systems
- A/B Testing
- App Configurations

Network Layer

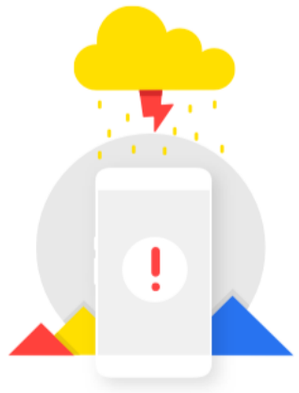
Requirements

Request Blocking



Requirements

- GET / POST JSON Requests + Blob Downloads
- Request Blocking
- Request Priority Based on Network Type (2G, 3G, 4G, Wi-Fi)
- Request Throttling per Domain/Host
- More *Transparency* to the Caller



Something wrong!

Please try again after sometime.

OK

Existing Implementation

Internal Library

dispatched to another thread

AFNetworking

NSURLSession

```objc
-(void)makeHTTPCallWithRequest:(NSURLRequest *)request
{
    [self performSelectorInBackground:@selector(makeHTTPCallWithRequest:) withObject:request];
}
```

When to use `performSelectorInBackground:`

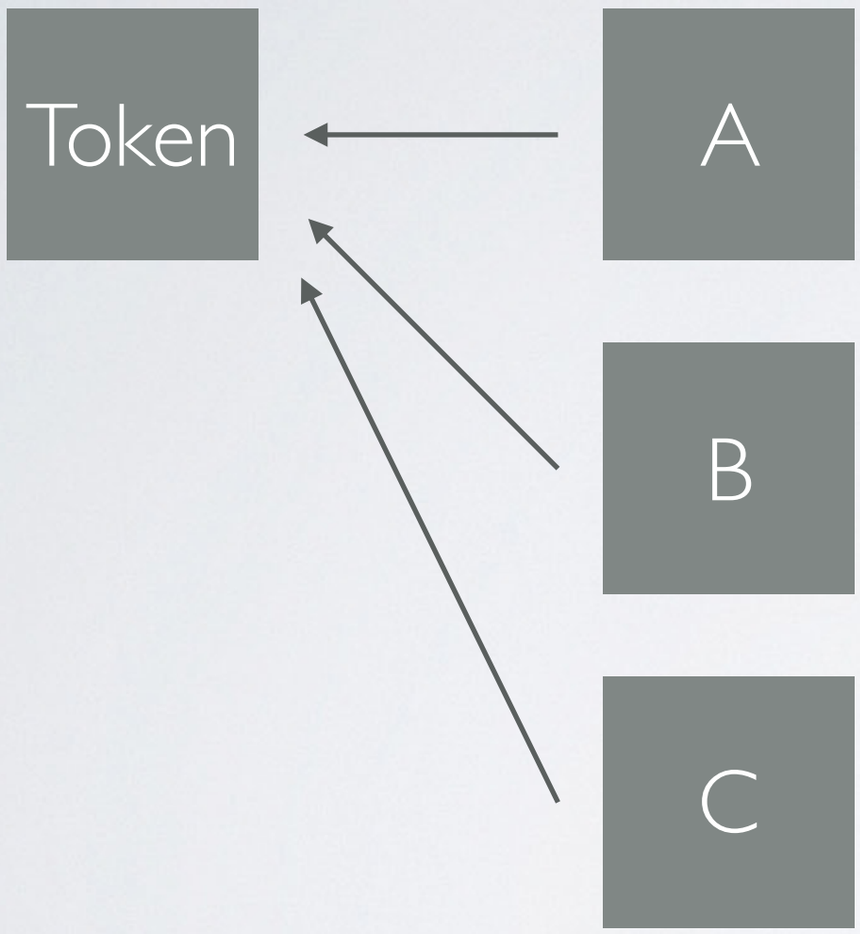
Never. Do not use this method. It spawns an unbounded number of threads. Even before GCD was available, this was a horrible method.

answered Oct 21 '13 at 18:02

Rob Napier

160k ● 21 ● 226 ● 341

Request Blocking



- Blocking Dictionary
- Array Holding the Requests
- Mutation Through Queues
- Key Value Observation (KVO) to Unblock

```
serialQueue = dispatch_queue_create(key, DISPATCH_QUEUE_SERIAL);
requestsBeingProcessedQueue = dispatch_queue_create(requestsBeingProcessedKey, DISPATCH_QUEUE_SERIAL);
```

Existing Performance

```
[self measureMetrics:@[XCTPerformanceMetric_WallClockTime] automaticallyStartMeasuring:true forBlock:^{

    XCTestExpectation *expectation = [self expectationWithDescription:@"[redacted]: GET"];

    request.successBlock = ^(NSURLSessionDataTask * operation, id responseObject){
        [expectation fulfill];
    };

    request.failureBlock = ^(NSURLSessionDataTask * operation, id responseObject){
        [expectation fulfill];
    };

    [[redacted] makeHTTPCallWithRequest:request];

    [self waitForExpectationsWithTimeout:8 handler: ^(NSError *error) {
        [self stopMeasuring];
    }];

}];
```

```
<unknown>:0: Test Case '-[DataTestProjectTests test[redacted]GET]' measured [Time, seconds] average: 1.248, relative standard deviation: 15.759%, values:
[1.409979, 1.147099, 1.124339, 0.988904, 1.152226, 1.331359, 1.733279, 1.227663, 1.121389, 1.245149],
performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime, baselineName: "", baselineAverage: , maxPercentRegression: 10.000%,
maxPercentRelativeStandardDeviation: 10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
Test Case '-[DataTestProjectTests test[redacted]GET]' passed (12.936 seconds).
```

[1.409979, 1.147099, 1.124339, 0.988904, 1.152226, 1.331359, 1.733279, 1.227663, 1.121389, 1.245149]

Average: 1.248 seconds

Requirements

- GET / POST JSON Requests + Blob Downloads 🙃
- Request Blocking 😐
- ~~Request Priority Based on Network Type (2G, 3G, 4G, Wi-Fi)~~
- ~~Request Throttling per Domain/Host~~
- ~~More *Transparency* to the Caller~~

It was time to create something better.

What's Available

ALAMOFIRE

```
Alamofire.request("https://httpbin.org/get").responseJSON { response in
    print(response.request)  // original URL request
    print(response.response) // HTTP URL response
    print(response.data)     // server data
    print(response.result)   // result of response serialization

    if let JSON = response.result.value {
        print("JSON: \(JSON)")
    }
}
```

Siesta

```
MyAPI.resource("/profile").addObserver(owner: self) {
    [weak self] resource, _ in

    self?.nameLabel.text = resource.jsonDict["name"] as? String
    self?.colorLabel.text = resource.jsonDict["favoriteColor"] as? String

    self?.errorLabel.text = resource.latestError?.userMessage
}
```

```
public enum HTTPMethod: String {
    case options = "OPTIONS"
    case get     = "GET"
    case head    = "HEAD"
    case post    = "POST"
    case put     = "PUT"
    case patch   = "PATCH"
    case delete  = "DELETE"
    case trace   = "TRACE"
    case connect = "CONNECT"
}
```

- HTTP Basic
- HTTP Digest
- Kerberos
- NTLM

Property List Encoding

URL Encoding

JSON Encoding

```
MyAPI.profile.request(.post, json: ["foo": [1,2,3]])
MyAPI.profile.request(.post, urlEncoded: ["foo": "bar"])
MyAPI.profile.request(.post, text: "Many years later, in front of the terminal...")
MyAPI.profile.request(.post, data: rawData, contentType: "text/limerick")
```

NSURLConnection

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

```
#pragma mark NSURLConnection Delegate Methods

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)r
esponse {
    // A response has been received, this is where we initialize the instance var you
 created
    // so that we can append data to it in the didReceiveData method
    // Furthermore, this method is called each time there is a redirect so reinitializ
ing it
    // also serves to clear it
    _responseData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    // Append the new data to the instance variable you declared
    [_responseData appendData:data];
}

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
                  willCacheResponse:(NSCachedURLResponse*)cachedResponse {
    // Return nil to indicate not necessary to store a cached response for this connec
tion
    return nil;
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    // The request is complete and data has been received
    // You can parse the stuff in your instance variable now

}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    // The request has failed for some reason!
    // Check the error var
}
```

```
// Create the request.
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:@"http://goog
le.com"]];

// Create url connection and fire request
NSURLConnection *conn = [[NSURLConnection alloc] initWithRequest:request
delegate:self];
```

NSURLSession

```objc
NSURL *URL = [NSURL URLWithString:@"http://example.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

NSURLSession *session = [NSURLSession sharedSession];
NSURLSessionDataTask *task = [session dataTaskWithRequest:request
                                        completionHandler:
    ^(NSData *data, NSURLResponse *response, NSError *error) {
        // ...
    }];

[task resume];
```

```swift
URLSession.shared.dataTask(with: request, completionHandler: { (data, response, error) in

}).resume()
```

`HTTPMaximumConnectionsPerHost`

The maximum number of simultaneous connections to make to a given host.

`ephemeralSessionConfiguration`

Returns a session configuration that uses no persistent storage for caches, cookies, or credentials.

`allowsCellularAccess`

A Boolean value that determines whether connections should be made over a cellular network.

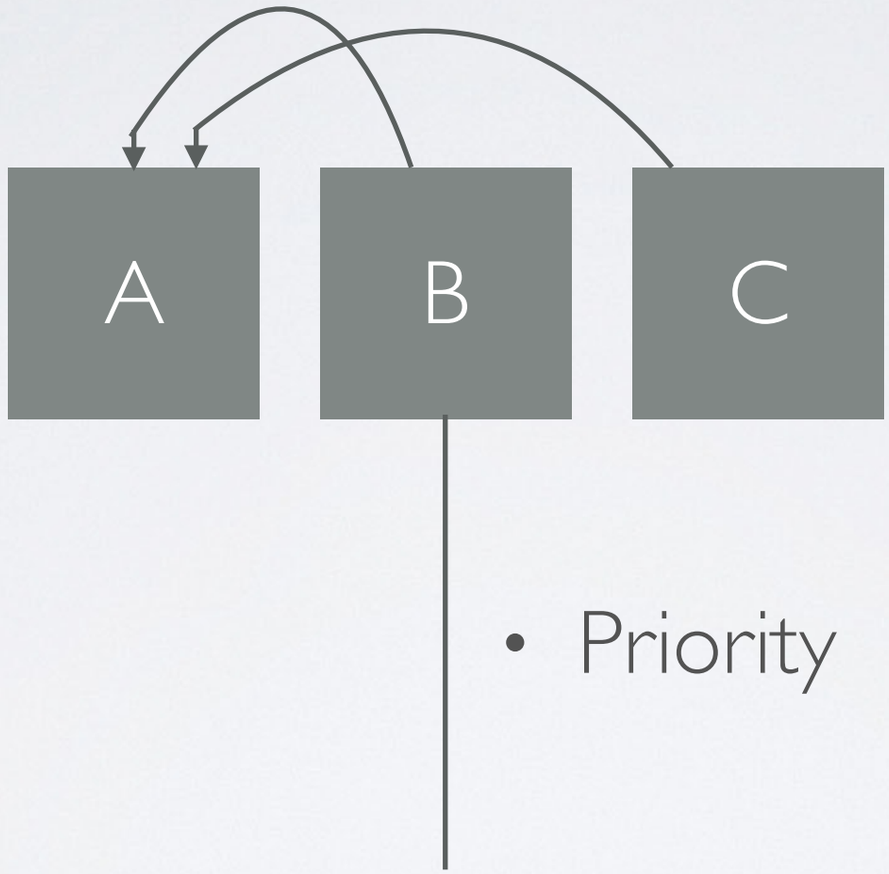
NSOperations & NSOperationQueue

Advanced NSOperations

Operations are a flexible way to model your app's business logic, but they can do so much more. See how NSOperation forms the heart of the WWDC app, and how using features like dependencies, readiness, and composition allow you to quickly and easily build dynamic and complex apps.

WWDC 2015 - Session 226 - iOS, macOS

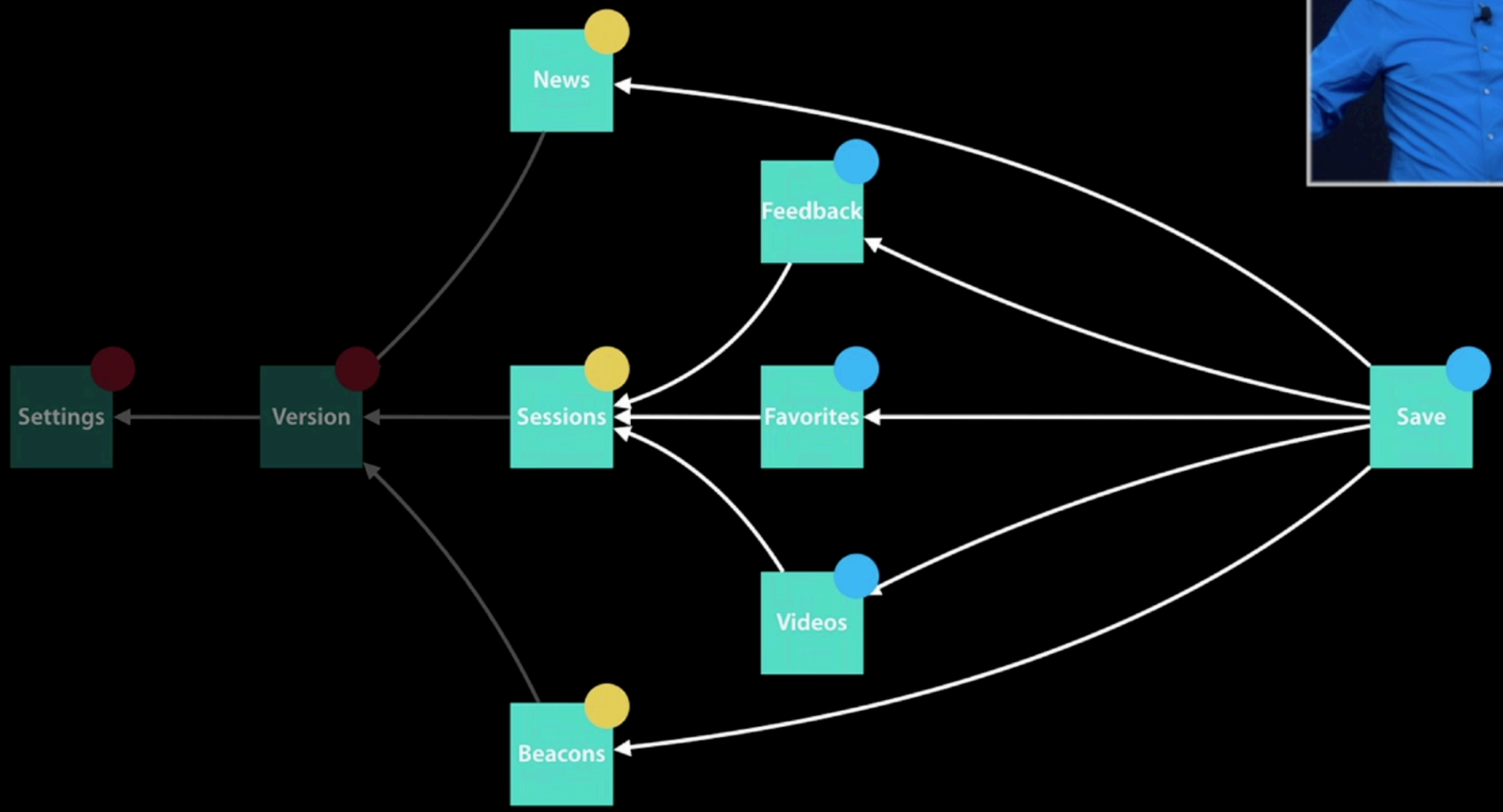
- Dependencies



- Priority

WWDC App Startup

Operation dependencies



16:32 -27:42

```
class MyCustomOperation: Operation {
    override func main(){
        //Make Network Request Here
    }
}
```

```swift
class MyCustomOperation: Operation {
    override func main(){

        //async Block
        URLSession.shared.dataTask(with: URL.init(string: "https://httpbin.org/get")!) { (data, response, error) in
            print("Request Finished with response: \(response)")
        }

        //End of main method - Operation Finishes Early
    }
}
```

```
import Foundation

class MyCustomOperation: Operation{
    var isExecuting = false;
    var isFinished = false;

    override func main() {
        if self.cancelled {
            return
        }
        else
        {
            //Make Network Request Here
        }
        self.willChangeValueForKey("executing")
        isExecuting = false
        self.didChangeValueForKey("executing")

        self.willChangeValueForKey("finished")
        isFinished = true
        self.didChangeValueForKey("finished")

        if(finished)
        {
            NSLog("Operation completed")
        }
        else
        {
            NSLog("Not completed")
        }
    }
}
```

arikis / **Overdrive**

Overdrive

Powerful task based Swift API with focus on type safety, concurrency and multi threading

Task<T>

override func run()

Call self.finish(.value(T)) when you're done

```
class GetLogoTask: Task<UIImage> {

  override func run() {
      let logoURL = URL(string: "https://swiftable.io/logo.png")!

      do {
          let logoData = try Data(contentsOf: logoURL)
          let image = UIImage(data: logoData)!
          finish(with: .value(image)) // finish with image
      } catch {
          finish(with: .error(error)) // finish with error if any
      }
  }
}
```

```
let logoTask = GetLogoTask()

logoTask
    .onValue { logo in
        print(logo) // UIImage object
    }.onError { error in
        print(error)
}
```

```swift
public struct SwiftyNetworkTaskResult {
    let operation: URLSessionDataTask
    let response : URLResponse?
    let data: Any?
    let error : Error?
}
```

```swift
public class SwiftyNetworkTask: Task <SwiftyNetworkTaskResult>
```

```swift
override public func run() {

    var dataTask : URLSessionDataTask?

    dataTask = urlSession.dataTask(with: self.request, completionHandler: { (data, response, error) in
        if let error = error {
            self.finish(with: .value(SwiftyNetworkTaskResult(operation: dataTask!, response: response, data: nil, error: error)))
        }
        else if let response = response, let data = data {
            self.parseAndFinish(dataTask: dataTask!, urlResponse: response, data: data)
        }
    })

    if let dataTask = dataTask {
        dataTask.resume()
    }
    else {
        let error = NSError.init(domain: "Swifty failed to create a successful URLSessionDataTask with the given params.", code: 1,
            userInfo: nil)
        self.finish(with: .error(error))
    }
}
```

```
public class SwiftNetworkManager: NSObject {

    public static let sharedInstance = SwiftNetworkManager()
```

```swift
public func addOperation(_ request: URLRequest, options: [SwiftyRequestOptions] = []){
```

```swift
let request = SwiftyNetworkTask(request: request)

request
    .onValue { (swiftyResponse) in

        print(swiftyResponse.response!)

}.onError { error in
    print(error)
}
```

```swift
self.networkQueue.add(task: request)
```

```swift
public func addOperation(_ request: URLRequest, options: [SwiftyRequestOptions] = []){
```

```swift
let request = SwiftyNetworkTask(request: request)

request
    .onValue { (swiftyResponse) in

        print(swiftyResponse.response!)

}.onError { error in
    print(error)
}
```

```swift
self.networkQueue.add(task: request)
```

```swift
public enum SwiftyRequestOptions {
    case isBlockable
    case priority(Operation.QueuePriority)
}
```

```swift
for option in options {
    switch(option){
    case .isBlockable:
        if let conditionManager = self.conditionManager {
            request.add(condition: conditionManager)
        }
    case .priority(let priority):
        request.queuePriority = priority
    default:
        continue
    }
}
```

```swift
self.networkQueue.add(task: request)
```

```swift
public enum SwiftyRequestOptions {
    case isBlockable
    case priority(Operation.QueuePriority)
}
```

```swift
for option in options {
    switch(option){
    case .isBlockable:
        if let conditionManager = self.conditionManager {
            request.add(condition: conditionManager)
        }
    case .priority(let priority):
        request.queuePriority = priority
    default:
        continue
    }
}
```

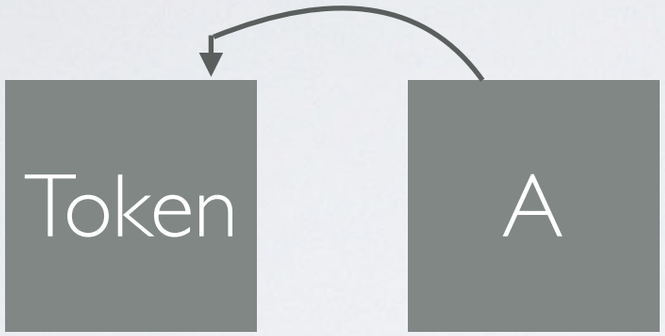
```swift
self.networkQueue.add(task: request)
```

```swift
public enum SwiftyRequestOptions {
    case isBlockable
    case priority(Operation.QueuePriority)
}
```

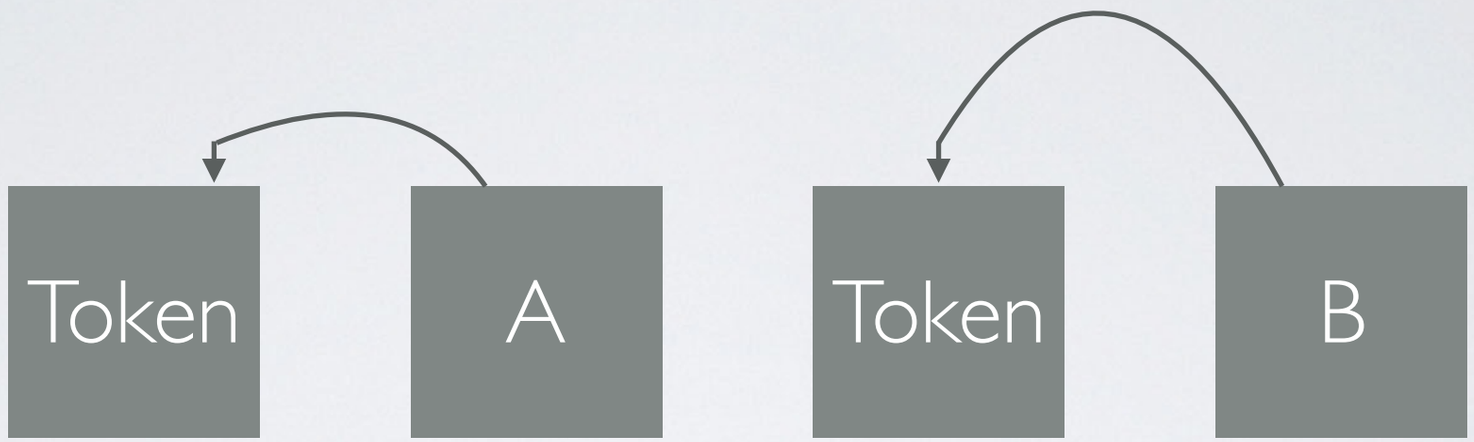
```swift
for option in options {
    switch(option){
    case .isBlockable:
        if let conditionManager = self.conditionManager {
            request.add(condition: conditionManager)
        }
    case .priority(let priority):
        request.queuePriority = priority
    default:
        continue
    }
}
```

```swift
self.networkQueue.add(task: request)
```

Queue:



Queue:



This is wrong. We're executing the Token Task Twice.

Queue:

But, A and B come from different callers,
and at different times T1, and T2.

Constraint Manager

1. Tells **A** to wait for **Token** to Finish

2. Returns the **Token** Task to add to queue

Tells **B** to wait for **Token** to Finish

Queue:

Token

A

B

```swift
static let shared = ConstraintManager()

var enqueuedTask = false

public func dependencies<T>(forTask task: Task<T>) -> [Operation] {

    var dependencies : [Operation] = []

    if(enqueuedTask){
        task.add(dependency: TokenTask.shared)
    }
    else {
        enqueuedTask = true
        dependencies.append(TokenTask.shared)
    }

    return dependencies
}
```

```swift
public enum SwiftyRequestOptions {
    case isBlockable
    case priority(Operation.QueuePriority)
}
```

```swift
for option in options {
    switch(option){
    case .isBlockable:
        if let conditionManager = self.conditionManager {
            request.add(condition: conditionManager)
        }
    case .priority(let priority):
        request.queuePriority = priority
    default:
        continue
    }
}
```

```swift
self.networkQueue.add(task: request)
```

Requirements

- GET / POST JSON Requests + Blob Downloads (URLSession) ✅😏
- Request Blocking (OperationQueue Dependencies + ConstraintManager) ✅😏
- Request Priority Based on Network Type (2G, 3G, 4G, Wi-Fi) (OperationQueue Priorities) ✅😏
- Request Throttling per Domain/Host (URLSession maxConnectionsPerHost) ✅😏
- More *Transparency* to the Caller (Return URLResponse to the caller) ✅😏

What about Performance?

Existing Performance

```
[self measureMetrics:@[XCTPerformanceMetric_WallClockTime] automaticallyStartMeasuring:true forBlock:^{

    XCTestExpectation *expectation = [self expectationWithDescription:@"[redacted]: GET"];

    request.successBlock = ^(NSURLSessionDataTask * operation, id responseObject){
        [expectation fulfill];
    };

    request.failureBlock = ^(NSURLSessionDataTask * operation, id responseObject){
        [expectation fulfill];
    };

    [[redacted] makeHTTPCallWithRequest:request];

    [self waitForExpectationsWithTimeout:8 handler: ^(NSError *error) {
        [self stopMeasuring];
    }];

}];
```

```
<unknown>:0: Test Case '-[DataTestProjectTests test[redacted]GET]' measured [Time, seconds] average: 1.248, relative standard deviation: 15.759%, values:
[1.409979, 1.147099, 1.124339, 0.988904, 1.152226, 1.331359, 1.733279, 1.227663, 1.121389, 1.245149],
performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime, baselineName: "", baselineAverage: , maxPercentRegression: 10.000%,
maxPercentRelativeStandardDeviation: 10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
Test Case '-[DataTestProjectTests test[redacted]GET]' passed (12.936 seconds).
```

[1.409979, 1.147099, 1.124339, 0.988904, 1.152226, 1.331359, 1.733279, 1.227663, 1.121389, 1.245149]

Average: 1.248 seconds

Existing Performance

```
measureMetrics([XCTPerformanceMetric_WallClockTime], automaticallyStartMeasuring: true) {
    let expec = self.expectation(description: "GET")

    var GETRequest : URLRequest {
        let randomNum:UInt32 = arc4random_uniform(100)
        var request = URLRequest.init(url: URL.init(string: "https://httpbin.org/get?number=\(randomNum)")!)
        request.httpMethod = "GET"
        return request
    }

    self.manager!.addOperation(GETRequest, options: [], successBlock: { (dataTask, data) in
        expec.fulfill()
    }, failureBlock: { (dataTask, error) in
        expec.fulfill()
    })

    self.waitForExpectations(timeout: 5) { error in
        if let error = error {
            print("Error: \(error.localizedDescription)")
        }
        self.stopMeasuring()
    }
```

```
/Users/siddharth.gupta/Desktop/SiestaBasedNetwork/Swifty/Example/Tests/Tests.swift:92: Test Case '-[Swifty_Tests.Tests testGET]' measured [Time, seconds] average: 0.315,
relative standard deviation: 19.535%, values: [0.249881, 0.448979, 0.307115, 0.306861, 0.307096, 0.306851, 0.306754, 0.219645, 0.394383, 0.306945],
performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime, baselineName: "", baselineAverage: , maxPercentRegression: 10.000%, maxPercentRelativeStandardDeviation:
10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
Test Case '-[Swifty_Tests.Tests testGET]' passed (3.518 seconds).
```

[0.249881, 0.448979, 0.307115, 0.306861, 0.307096, 0.306851, 0.306754, 0.219645, 0.394383, 0.306945]

Average: 0.315 seconds

75% Decrease in Time

~3.9x the performance boost
of the previous implementation

```
Test Case '-[Swifty_Tests.Tests testGET_Alamofire]' started.
/Users/siddharth.gupta/Desktop/SiestaBasedNetwork/Swifty/Example/Tests/Tests.swift:180: Test Case '-[Swifty_Tests.Tests testGET_Alamofire]' measured [Time, seconds] average:
0.476, relative standard deviation: 120.555%, values: [2.193620, 0.212774, 0.297354, 0.306658, 0.306249, 0.307398, 0.306565, 0.307091, 0.306904, 0.214404],
performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime, baselineName: "", baselineAverage: , maxPercentRegression: 10.000%, maxPercentRelativeStandardDeviation:
10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
Test Case '-[Swifty_Tests.Tests testGET_Alamofire]' passed (5.067 seconds).
Test Case '-[Swifty_Tests.Tests testGET_URLSession]' started.
/Users/siddharth.gupta/Desktop/SiestaBasedNetwork/Swifty/Example/Tests/Tests.swift:138: Test Case '-[Swifty_Tests.Tests testGET_URLSession]' measured [Time, seconds] average:
0.314, relative standard deviation: 7.057%, values: [0.380866, 0.306856, 0.306823, 0.306744, 0.306837, 0.306812, 0.306813, 0.306201, 0.307549, 0.307847],
performanceMetricID:com.apple.XCTPerformanceMetric_WallClockTime, baselineName: "", baselineAverage: , maxPercentRegression: 10.000%, maxPercentRelativeStandardDeviation:
10.000%, maxRegression: 0.100, maxStandardDeviation: 0.100
Test Case '-[Swifty_Tests.Tests testGET_URLSession]' passed (3.471 seconds).
```

Alamofire: ~ 0.476 seconds

URLSession: ~ 0.314 seconds

Swifty: ~ 0.315 seconds

Takeaways

- Try to figure out the exact requirements, Design accordingly
- If you find a library that meets those requirements perfectly, great!
- Always take a serious look at Apple APIs (WWDC Videos are a great resource)
- Read the code of Open Source Libraries, take advantage of the 'open source' part.
- Try and try to simplify until it 'feels right'

Thank You