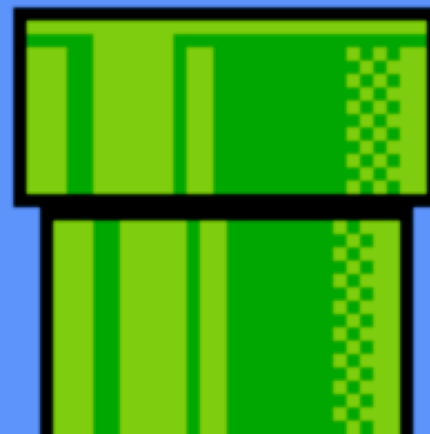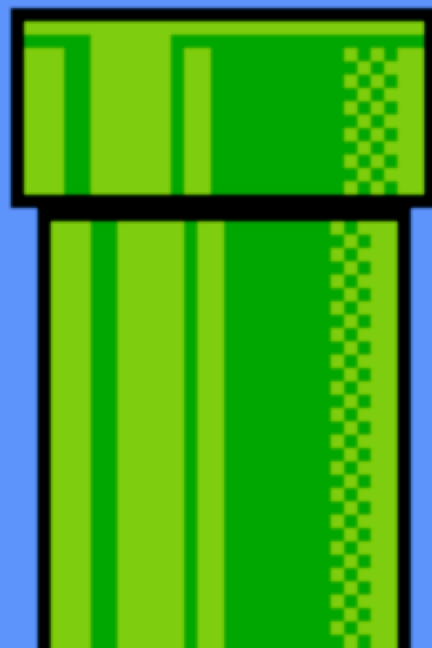# LEVELING UP WITH PIPELINES

# WHAT ARE PIPELINES?

# QUICK EXAMPLE

```
cat logs.txt | grep "ERROR" | wc -l
```

# BENEFITS OF PIPELINES

» Divides your app into small chunks

» Stages executed in series

» Stages can be reused in different pipelines

» Adds readability

» Stages follow SRP

» Smaller, more testable code

EXAMPLES

MIDDLEWARE

# LARAVEL

```php
// Within App\Http\Kernel Class...

protected $routeMiddleware = [
    'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
];
```

```php
class CheckAge
{
    public function handle($request, Closure $next)
    {
        if ($request->age <= 200) {
            return redirect('home');
        }

        return $next($request);
    }
}
```

ECOMMERCE

# ECOMMERCE

```php
$order = new Order;
```

```php
$order = new Order;

$paymentPipeline = (new Pipeline)
    ->pipe(new ApplyCoupons)
    ->pipe(new ApplyTaxes)
    ->pipe(new AddShipping)
    ->pipe(new ProcessPayment);
```

# ECOMMERCE

```php
$order = new Order;

$paymentPipeline = (new Pipeline)
    ->pipe(new ApplyCoupons)
    ->pipe(new ApplyTaxes)
    ->pipe(new AddShipping)
    ->pipe(new ProcessPayment);

$orderPipeline = (new Pipeline)
    ->pipe(new CreateOrder)
    ->pipe($paymentPipeline)
    ->pipe(new SendInvoice);

$orderPipeline->process($order);
```

# NOTIFICATIONS

## NOTIFICATIONS

```php
$notification = new Notification;

$notificationPipeline = (new Pipeline)
    ->pipe(new Notifications\Web)
    ->pipe(new Notifications\Mobile)
    ->pipe(new Notifications\Email);

$notificationPipeline->process($notification);
```

REFACTORING

```php
// Get Latest Activity (email events, associated engagements)
if ($this->shouldFetchRelatedDataType(SomeFakeCrmApp::KEY_LATEST_ACTIVITY, $objectType)) {
    $activityPromises = [
        'engagements' => $this->provider->getAssociatedEngagementsForObject($objectType, $objectId)
    ];

    $activityPromise = \GuzzleHttp\Promise\all($activityPromises)->then(function(array $responses){
        return new FulfilledPromise($this->zipAndSortLatestActivity($responses));
    });

    $promises[SomeFakeCrmApp::KEY_LATEST_ACTIVITY] = $activityPromise;
}

// Get Lists
if ($this->shouldFetchRelatedDataType(SomeFakeCrmApp::KEY_LISTS, $objectType)) {
    $promises[SomeFakeCrmApp::KEY_LISTS] = $this->provider->getLists(
        array_pluck(array_get($profile, 'lists', []), 'static-list-id')
    );
}

// Get Workflows
if ($this->shouldFetchRelatedDataType(SomeFakeCrmApp::KEY_WORKFLOWS, $objectType)) {
    $promises[SomeFakeCrmApp::KEY_WORKFLOWS] = $this->provider->getWorkflowsForContact($objectId);
}

// Get Deals
if ($this->shouldFetchRelatedDataType(SomeFakeCrmApp::KEY_DEALS, $objectType)) {
    $promises[SomeFakeCrmApp::KEY_DEALS] = $this->provider->getDealsForObject($objectType, $objectId);
}

return \GuzzleHttp\Promise\unwrap($promises);
```

```php
$relatedDataPipeline = new RelatedDataPipeline(
    $this->provider,
    $profile
);


return $relatedDataPipeline->processPipeline();
```

```php
$stages = [
    SomeFakeCrmApp::KEY_LATEST_ACTIVITY  => LatestActivityPipeline::class,
    SomeFakeCrmApp::KEY_LIST_MEMBERSHIPS => ListsStage::class,
    SomeFakeCrmApp::KEY_WORKFLOWS        => WorkflowsStage::class,
    SomeFakeCrmApp::KEY_DEALS            => DealsStage::class,
];

function buildPipeline() {
    $builder = new PipelineBuilder;

    foreach ($this->stages as $stageKey => $stage) {
        if ($this->shouldAddStage($stageKey) === false) {
            continue;
        }

        $builder->add(new $stage);
    }

    $this->pipeline = $builder->build();
}

function processPipeline(): array {
    $promises = $this->pipeline->process([]);
    return \GuzzleHttp\Promise\unwrap($promises);
}
```

# REUSING PIPELINES

```php
class LatestActivityPipeline {
    protected $stages = [
        AssociatedEngagementsStage::class,
        EmailEventsStage::class
    ];
}
```

```php
function buildPipeline() {
    $stages = collect($this->stages)
        ->filter(function($stage, $stageKey){
            return $this->shouldAddStage($stageKey);
        })
        ->map(function($stage){
            return new $stage;
        })
        ->toArray();

    $this->pipeline = new Pipeline($stages);
}
```

# BENEFITS - A RECAP

» Smaller, more managable stages

» Reusable stages

» Increased readability

» Better testing!

MARIO
013300 ×16

WORLD
1-1

TIME
229

5000

# REFERENCES

» Design Pattern: the Pipeline

» The Pipeline Pattern — for fun and profit

» How to use the Pipeline Design Pattern in Laravel

» League\Pipeline