Scout APM

New Relic Comparison Guide

Table of Contents

01	App Overview	
02	Language Support & Prici	ing
03	Transaction Traces	
04	Web Endpoints	
05	Web Endpoint Detail	
06	Database Monitoring	
07	Background Jobs	
08	Development Profiler	
09	Weekly Trends Email	
10	Agent, Alerting & Error A	nalytics
11	Conclusion	



Scout vs New Relic

There's no type of monitoring tool that can get to the heart of a performance problem faster than Application Performance Management (APM).

If you're making this decision, you may be choosing between New Relic and <u>Scout</u>. There are cases where New Relic is a better fit, clear cases where Scout fits like a glove, and some murky situations as well.

Where do you fall?

In this guide, we will summarize the factors to consider when choosing your best APM product.

CHAPTER 1 App Overview

This is the first page you see when checking the health of your app. Scout's version is below and New Relic's on the following page:



New Re	elic Ap	יO ק	vervie	ew							
New Relic.	~					Maps Ale	rts ^{New} Tools	∽ Help √	💓 app4856	53282@heroku	1 ~
Applications Service maps	Key transactions Alerts										
APPS ProjectPlanner.io	TIME PICKER Last 6 hours ending	now 💛	SERVERS ~								
MONITORING	Web transaction	ons time	~			295 m	s 7.1 s	Apdex so	:ore	.92 [0.5] 0	.84 [7.0]
Overview	450 ms							1		-	
Service maps	400 ms							0.5	W. III	W	M
Transactions	300 ms							0.25			
Databases	250 ms	MMA						Cadllees	Ann Conver		
External services	200 ms							End User	App Server		
Instances	150 ms										
Ruby VMs	100 ms							Through	put		3.09k rpm AVERAGE
	50 ms							3000			
Events	0					1.000		2000			
Errors	Request Queuing	GC Execution	Middleware	Ruby Post	gres Redis	Web extern	nal	1000			
Alerts								3 AM 9:00 A	M 10:00 AM11:00	0 AM12:00 PM 1	:00 PM
Deployments											
Thread profiler	Compare with	esterday and	d last week								
in cau promer	Transactions		App server time	e Error ra	te		0.00%	Member	of policy Defau	It application a	alert
REPORTS	IntercomSync/exe	cute	3.000 sec	LITOTIC			0.00%	policy			
SLA	Transaction traces:	4,820 s 4,700	s 3,940 s					Pere	nt events		
Availability	CheckinController	#checkin	313 ms							,	
Capacity	CheckinController	#ann server	load 191 ms					All 🗳	∥⊗ △	2 2 0	<>
Scalability	Transaction traces:	4.3 s 3.7 s 3	.5 s	0				(C) -DI]		
Web transactions	PlanTransactions	EnforcementJ	ob/ex 18.7 ms	D AM 9:00	AM 10:00 AM11:0	0 AM12:00 P	'M 1:00 PM	Yester	day		
Database	TraceAnalysisIob/	n/a	12 7 ms					APM			
Background jobs	Transaction traces:	n/a	12.7 113					Apdex s	core < 0.85		10:05
SETTINGS	8 hosts						:= 🖂	Norech	ical problem		10.00
Application	Host name	Apdex 🗘 R	esp. time 🗘 Thi	roughput 🗘 E	rror Rate 🗘 CF	PU usage 🔇	Memory O	This W	eek		
Availability monitoring	in 40.4.0.425							APM Andex s	core < 0.85		
Environment	32 app instances	0.92 _{0.5}	288 ms	608 rpm	0.00 %	215 %	18 GB	Non-crit	tical problem	8	:33 Wed
								APM Andex s	core < 0.85		
ALERTS >	ip-10-1-0-202 32 app instances	0.99 _{0.5}	76.5 ms	95 rpm	0.00 %	26 %	14 GB	Non-crit	tical problem	1	0:03 Tue
								APM Apday s	coro < 0.85		
	ip-10-1-0-243 32 app instances	0.91 _{0.5}	316 ms	568 rpm	0.00 %	233 %	18 GB	Non-crit	tical problem		9:03 Tue
								APM	. 0.05		
	ip-10-1-0-76	0.9205	298 ms	596 rpm	0.00 %	217 %	18 GB	Apdex s	core < 0.85 tical problem		7:21 Tue
	32 app instances	0.5						APM			
	ip-10-1-1-28	0.99	77.6 ms	95 rom	0.00 %	3E 04	1/ CP	Apdex s Non-crit	core < 0.85 tical problem	8:4	41 10/09
	32 app instances	0.590.5	77.01115	99 (hui	0.00 %	20 %	14 00	APM			
	ip-10-1-1-89							Apdex s Non-crit	core < 0.85 tical problem	7:	36 10/09
	32 app instances	0.91 _{0.5}	321 ms	566 rpm	0.00 %	205 %	17 GB				

The top portion of the page is similar between New Relic and Scout: a breakdown of time spent by category (ex: Ruby, Database, External HTTP services, etc) over time. You can view data across similar timeframes in both Scout and New Relic (New Relic offers three months of data in their Pro package and Scout can do the same in their custom plans).

Both let you compare data to the past. Scout gives a few more options than New Relic, which is restricted to yesterday and last week comparisons. The biggest difference with Scout's approach: several algorithms dig through your data and generate insights on performance directly on this page. For example, Scout identifies <u>slow database queries</u>, <u>N+1</u> <u>database queries</u>, and <u>memory bloat</u>. These are ordered by impact, which help you decide where to get most improvements in the least amount of time.

Scout short-circuits the work a Site Reliability Engineer (SRE) often performs when investigating the health of an app.

Let's move from the 10,000 foot view to the soul of application monitoring.

CHAPTER 2 Language Support & Pricing

Scout supports Ruby, Elixir, PHP, Node.js, and Python with a Core Agent API that can be used to instrument any language. New Relic supports Ruby, Java, Node.js, PHP, .NET, Python, and Go.

Scout's monthly fee is about 30% lower than the monthly fee in a New Relic annual contract. In most cases, Scout will also buy your New Relic contract.

CHAPTER 3 Transaction Traces

The soul of app monitoring is the transaction trace: all the metrics and analysis an APM platform delivers originates from data collected in a trace. A transaction trace is a breakdown of time spent in a single web request or execution of a single background job.

A Scout trace is below and New Relic is on the next page:

Applicat	tions	*		#	slack	👩 Roadmap	Docs (Blog Use	r 🕐 🖌
ProjectPlann	er.io 🖗 🔤				٩	Last 7 days endir	ng now 😽	Compare To	~
Overview	b Endpoints 🛛 🗸	Background Jobs	✓ Database	Traces Error	a Aler	ts Reports	Settings	Metadata	C
← ServicesCo	ontroller#in	dex							
© Mon 4:21:11 P	M · /apps/1/se	rvices							
Time Breakdown	Memory Alloca	tion Breakdown	Context						
Posponso Timo	ActiveP	acord	InfluxDB	View		ЦТТР	0	ther	
2.732 m	1S 41%	ecolu	40%	11%		5%	3	3%	
_,	298 calls		E calle	57 calls					
	2.50 Calis	2	J Calls	57 Calis		T Calls			
Summary	eline Condens	sed View							
		Midd	lleware						
		Router	/Rails						
		ServicesController	#index						
		User#find	SQL						
		App#fin	d SQL						
		Membership#fin	d SQL						
		Org#fin	d SQL						
	SQL#find	BACKTRACE • 267 cal	ls 时						
- 🧾 app/models/	org.rb in influx	_read_router at lin	ne 233						
230 def	fault = influx_r .readable	outers							
232 .	includes(:influx_	server)					See th	ne line-c	of-
233	first		•				code	and auth	or
234 end							couc		.01
+ app/models/	org.rb in influx	_read_client at lin	ne 252						
+ Rapp/models/	org.rb in influx	at line 270							
+ 📑 lib/queries	/db_metrics/all.r	b in call at line	62						
+ m lib/queries	/db_metrics/all.r	b in values at lin	ne 51						
+ map/models/	presenters/servic	es/overview.rb in t	otal_time_consumed	at line 80					
+ pp/models/	presenters/servic	es/overview.rb in c	lock in generate_qu	eries at line 74					
Man app/models/	presenters/servic	es/overview.rb in e	ach at line os						
	Provisioning	app#Tin g::HerokuInstall#fin	d SQL						
AgentSupport.from	_app(cup).data	base_monitoring?	ACKTRACE						
	SQL#find SQL	BACKTRACE • 6 Ca	lls 111					I	
		AppMetadata#fir	nd SQL						
c	urrent_org.addons	.active(:db).any?	BACKTRACE						
		SQL#other SQL	• 2 calls						
		InfluxRouter#	find SQL						
		InfluxDB/Oue	y DETAILS						

New Re													
►													
PM ⁻	/						Maps	Alerts ^{New}	Tools 🗸	Help 🗸	əр 🞀	op48563282@herok	u 🚺
olications Service maps Ke	ey transa	actions Alerts											
°S M	TIME PI Last 6	ICKER 6 hours ending	now	SERVERS All servers	~								
	~	-											
IONITORING	Туре	Transac	tion trad	ce						Tr	ack as Ke	y Transaction	• 🛛
Overview	Mo	ip-10-1-1-92(v	vebrick:APM)										
Service maps		👻 File a ticke	t with JIRA 7	Delete this	s trace								
Transactions	Che	Oct 14, '17	12:13 pm	9,880 ms	5.570 m	s (56.4%) 71	.1 ms (0.72%)						
Databases	Che	TRACE TIME		RESP. TIME	CPU BURN	GC	TIME						
External services	Dep	Summary	Trace deta	ils Map ^{Be}	ta Databa	ase queries	Messages						
Instances	Aler												
Ruby VMs	Stat												
/ENITS	Арр												
Error application	App	0											
	Durk	U 1	0 %	20 %	20.01	10.01	50 %	60.%	70 9	6	80 %	90 %	100 5
Errore	PUC			20 /8	30 %	40 %	50 %	00 10					
Errors	Api:	CheckinContro	oller#checkin	Apm::Applica	30 %	40 % ActionDispatch:	:ParamsParser#c	ActionE	lispatch::Rout	ing::RouteSe	et#call	Postgres SlowJob i	insert
Errors Alerts	Api:	CheckinContro Postgres selec	oller#checkin t Remainde	Apm::Applica	30 %	40 % ActionDispatch:	:ParamsParser#c	all ActionE	Dispatch::Rout	ing::RouteSe	et#call	Postgres Slowjob i	insert
Errors Alerts Deployments	Api: Org	CheckinContro Postgres select	oller#checkin t <mark>Remainde</mark>	Apm::Applica	30 %	40 % ActionDispatch:	:ParamsParser#c	all ActionE	Dispatch::Rout	ing::RouteSe	et#call	Postgres Slowjob i	insert
Errors Alerts Deployments Thread profiler	Api: Org Hea	CheckinContro Postgres select	oller#checkin t Remainde S	Apm::Applica er	su %	40 %	:ParamsParser#c	ActionE	Dispatch::Rout	ing::RouteSe	et#call	Postgres Slowjob i Duration	insert %
Errors Alerts Deployments Thread profiler EPORTS	Api: Org Hea End	CheckinContro Postgres select Category Controller	soller#checkin Remainde S	Apm::Applica er lowest comp	tion#call v onents	40 % ActionDispatch:	Son ParamsParser#c	all ActionE	Dispatch::Rout	ing::RouteSe Coun	et#call	Postgres Slowjob i Duration 4,250 ms	insert % 43%
Errors Alerts Deployments Thread profiler EPORTS SLA	Api: Org Hea End	CheckinContro Postgres select Category Controller Rack App	ster verster verst	Apm::Applice	tion#call vonents	40 % ActionDispatch:	ParamsParser#c	ali ActionE	Dispatch::Rout	Coun	et#call	Postgres Slowjob i Duration 4,250 ms 4,190 ms	insert % 43% 42%
Errors Alerts Deployments Thread profiler EPORTS SLA Availability	Api: Org Hea End End App	CheckinContro Postgres selec Category Controller Rack App Middleware	ster Wcheckin t. Remaind S C A A	Apm:Applica tr lowest comp heckinContro pm::Application	tion#call vonents	40 % ActionDispatch: ser#call	ParamsParser#c	ActionE	Dispatch::Rout	Coun	et#call it 1 1	Postgres Slowjob I Duration 4,250 ms 4,190 ms 534 ms	insert (% 43% 42% 5%
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity	Api: Org Hea End End App End	CheckinContro Postgres select Category Controller Rack App Middleware Middleware	t Remainde	Apm::Applica ar lowest comp heckinContro .pm::Application .ctionDispatch .ctionDispatch	onents iller#checkin on#call :::ParamsPar :::Routing::Rc	40 % ActionDispatch: ser#call puteSet#call	.ParamsParser⊮c	ActionE	Dispatch::Rout	ing::RouteSe Coun	et#call it 1 1 1 1 1	Postgres Slowjob I Duration 4,250 ms 4,190 ms 534 ms 221 ms	insert %
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability	Api: Org Hea End End App End Wo	CheckinContro Postgres select Category Controller Rack App Middleware Database	t Remainds S C A A A P	Apm:Applica Iowest comp heckinContro pm::Application ctionDispatch ostgres Slowji	30% tionifcali onents liler#checkin on#call ::ParamsPar ::Routing::Rc ob insert	40 % ActionDispatch: ser#call puteSet#call	ParamsParser#c	Action:)ispatch::Rout	Coun	et#call it 1 1 1 1 0 0	Postgres Slowjob I Duration 4,250 ms 4,190 ms 534 ms 221 ms	insert %
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions	Api: Org Hea End App End Wor Gitt	CheckinContro Postgres select Category Controller Rack App Middleware Middleware Database Database	Solier#checkin Remainder S C C A A A A A P P P P	Apm:Application of the second	30% tion#call conents ller#checkin on#call ::ParamsPar ::Routing::Rc ob insert t	40 % ActionDispatch: ser#call puteSet#call	ParamsParser//c	all Action)ispatch::Rout	Coun	et#call	Duration 4,250 ms 4,190 ms 534 ms 221 ms 219 ms	insert % 43% 42% 2% 2% 2%
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database	Api: Org Hea End End App End Woi Gitt Dev	CheckinContro Postgres select Category Controller Rack App Middleware Middleware Database Database Remainder	Remainds Remainds S C C A A A A A A P P P P P R	Apm:Applica Apm:Applica Iowest comp heckinContro pm::Application ctionDispatch ostgres Slowj ostgres select emainder	onents Unr#call Unr#checkin on#call Unr#checkin on#call UnramsPar Cob insert t	40 % ActionDispatch: ser#call buteSet#call	200)ispatch::Rout	Coun	et#call	Postgres Slowjob I Duration 4,250 ms 4,190 ms 534 ms 221 ms 219 ms 153 ms 313 ms	insert 43% 42% 2% 2%
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background jobs	Api: Org Hea End End App End Gitt Gitt Dev	CheckinContro Postgres selec Category Controller Rack App Middleware Diddleware Database Remainder	Remainds Remainds C A A A A A A A A A A A A A A A A A A	Apm:Applica Apm:Applica Iowest comp heckinContro pm:Application ctionDispatch ostgres Slowj ostgres select emainder	onents iller#checkin on#call :::ParamsPar :::Routing::Rc ob insert :	40.% ActionDispatch	200)ispatch::Rout	- ing::RouteSe Coun - - - - - - - - - - - - - - - - - - -	et#call 1 1 1 1 0 0 1	Postgres Slowjob Duration 4,250 ms 4,190 ms 534 ms 221 ms 219 ms 153 ms 313 ms 0 0000 ms	insert 99 4399 4299 299 299 399
Errors Alens Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background Jobs	Api: Org Hea End End End Gitt Dev Stri End	CheckinContro Postgres selec Category Controller Rack App Middleware Database Database Remainder	Remainder Remainder C C A A A A P P R R	Apm:Application Apm:Application Applicat	oonents onents iller#checkin on#call ::ParamsPar ::Routing::Rc ob insert t	40 % ActionDispatch	200	Action ()ispatch::Rout	- ing::RouteSe Coun - - - - - - - - - - - - - - - - - - -	et#call tt 1 1 1 0 0 1	Pargene Slowgeb I Duration 4,250 ms 4,190 ms 534 ms 221 ms 219 ms 153 ms 313 ms 9,880 ms	insert 43% 42% 23% 22% 33% 100%
Errors Alens Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background jobs ETTINGS	Api: Org Hea End App End App Gitt Dev Stri End Shov	CheckinContro Postgres selec Controller Rack App Middleware Middleware Database Database Remainder Total time URL	K Remainder Remainder C A A A P P P R	Apm:Application Apm:Application Applicat	30 % conents conents ller#checkin on#call :::ParamsPar :::Routing::Rc ob insert :: cckin.scout	40 % Accordispatchi Ser#call puteSet#call	2003		Sispate h::Rout	- ing::RouteSe Coun	et#call t t t t t t t t t t t t t t t t t t	Paragrees Slowyab J Duration 4,250 ms 4,190 ms 534 ms 221 ms 221 ms 153 ms 313 ms 9,880 ms	insert 43% 42% 2% 2% 3% 100%
Errors Alerts Deployments Thread profiler EFORTS SLA Availability Capacity Scalability Web transactions Database Background jobs EFTINGS Application	Api: Org Hea End App End Gitt Dev Stri End Shov	Checkincontrol Postgres selec Category Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio	Intersteheckin Remained S C C A A A P P R R n attribute	Apm:Application	30 % ition#call if ition#call	40 % ActionDispatchi Ser#call puteSet#call	2003	Actiont	Dispatch::Rout	- ing::RouteSi Coun - - - - - - - - - - - - - - - - - - -	et#call	Paragrees Slowyab J Duration 4,250 ms 4,190 ms 534 ms 2221 ms 2219 ms 153 ms 313 ms 9,880 ms	insert 43% 42% 2% 2% 2% 3% 100%
Errors Alerts Deployments Deployments Thread profiler EFORTS SLA Availability Capacity Scalability Web transactions Database Background jobs ETTINGS Application Availability monitoring	Api: Org Hee End App End Gitt Dev Stri End Shov	CheckinControl Postgres select Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio httpResponse	Intersteheckin Remained S C C A A A A P P R R n attribute eCode	Apm:Application	30% connects connects connects connects connection conn	40 % ActionDispatchs	2003	Action)ispatch::Rout	Coun Coun	et#call it it it i i i i i i i i i i i i i i	Paragrees Slowyddil Duration 4,4250 ms 4,4190 ms 534 ms 2211 ms 2219 ms 1533 ms 313 ms 9,880 ms	% 43% 42% 22% 22% 23% 30%
Errors Alerts Deployments Deployments Thread profiler EFORTS SLA Availability Capacity Web transactions Database Background jobs ETTINGS Application Availability monitoring Environment	Api: Org Hea End End App End Gitt Dev Stri End Shov	CheckinControl Postgres select Category Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio httpResponsi	Nerercheckin t Remaindr S C A A A A P P R R n attribute eCode	Arm:Application Arm:Application heckinContro pm:Application ctionDispatch ctionDispatch ostgres Slowj ostgres select emainder /apps/che ss (2)	30% conents confical iller#checkin oon#cal i::ParamsPar ::Routing::Rc ob insert t c cccut 200	40 % ActionDispatchs	200	all Actiont	lispatch::Rout	Coun	ett cett cett cett cett cett cett cett	Paragrees Slowyddal Duration 4,4,250 ms 4,4190 ms 534 ms 221 ms 2219 ms 1533 ms 313 ms 9,880 ms	**************************************
Errors Alerts Deployments Deployments Thread profiler EFORTS SLA Availability Capacity Scalability Web transactions Database Background jobs EFTINGS Application Availability monitoring Environment	Api: Org Hea End End App End Woi Gitt Dev Stri End Shov	CheckinControl Postgres select Category Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio httpResponsi	Nerercheckin t Remainds S C A A A P P R R n attribute eCode	Arm:Application Arm:Application Interview (Application) Interview (Application) Interview (Application) Armonomy (Application) Interview (Application) Armonomy (Application) Interview (Application)	30% stonifali st	40% AttionDispatcht sert#call uuteSet#call	PlaransPaneer (c	6:	Dispatch::Rout	Coun	et#call it it i i i i i i i i i i i i i i i i	Partgres Slowjobi Duration 4,250 ms 534 ms 221 ms 2219 ms 1533 ms 313 ms 9,880 ms	**************************************
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background Jobs ETTINGS Application Availability monitoring Environment EXERTS >	Api: Org Hea End App End Gitt Dev Stri End Shov	CheckinControl Postgres select Category Controller Rack App Middleware Middleware Database Database Remainder Total time URL Transactio httpResponse	Remainds C Remainds C C A A A A P P R R R R R R R R R R R R R	Arm:Application	30% stonfcall stonfcall stonfcall stonfcall stonfcall storfcall st	40 % Action Dispatcht ser#call buteSet#call contel contel	ParamstParaer / c	Actiont all Actions 65 all	Solopate-Root	Coun Coun 10 20 0	etterali it 1 1 1 1 0 0 1	Partgres Slowjobi Duration 4,250 ms 534 ms 221 ms 2219 ms 1533 ms 313 ms 9,880 ms	**************************************
Errors Alerts Deployments Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background Jobs ETTINGS Application Availability monitoring Environment ELERTS	Api: Org Hea End App End Woi Gitt Dev Stri End Shov	CheckinControl Postgres select Controller Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio httpResponse	sleerecheckin Remaindr S C C A A A P P P R R N R R N R R R R R R R R R R R R R	Arm:Application	su % suconfeati suconf	40 % Action Dispatch ser#call puteSet#call contex contex host	PPrantsPreef	65 aj	Jispatch-Route Solitation/js Job Jan Jinternal	Coun	ettreall 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	Partgres Slowgeb I Duration 4,4250 ms 534 ms 221 ms 2219 ms 153 ms 313 ms 9,880 ms	9% 43% 42% 55% 22% 22% 33% 100%
Errors Alerts Deployments Thread profiler EPORTS SLA Availability Capacity Scalability Web transactions Database Background jobs ETTINGS Application Availability monitoring Environment EERTS >	Api: Org Hea End App End Woi Gitt Dev Stri End Shov	Checkincontrol Postgres select Category Controller Rack App Middleware Middleware Database Remainder Total time URL Transactio httpResponse	steretcheckin t Remainds S C C A A A A A A P P R R R R R A A A A A A A A A A A A A	Arm:Applicati lowest comp heckinContro pm:Applicati ctionDispatch ctionDispatch ostgres Slowj ostgres select emainder /apps/che s:	su % onents connents	40.% ActionDispatch puteSet#call puteSet#call conter host userA	DParamstPaneer (C	Actiont 66 aj aj G	Separt - Sout State - Sout Stat	Coun	com	Partgres Slowgeb I Duration 4,4250 ms 534 ms 221 ms 2219 ms 153 ms 313 ms 9,880 ms	% 43% 43% 5% 2% 2% 3% 3% 3% 100%

There are some differences here as well:

- Scout provides a breakdown of memory allocations in addition to timing metrics. New Relic does not.
- New Relic offers cross-application tracing if your app touches another app you've monitored with New Relic. Scout does not.
- Scout is more likely to provide backtraces to slow method calls and can also display the code directly in the browser.
- Scout indicates the number of rows returned by SQL queries
- Scout can breakdown the time spent in custom code via ScoutProf. New Relic requires custom instrumentation.

Scout typically collects more traces than New Relic. Both New Relic and Scout have similar, one-week retention periods for traces.

chapter 4 Web Endpoints

Both New Relic and Scout let you view summary data on web endpoints in list format. Scout is listed first, New Relic second:

Scout Web Endp	oints				
•					
Contractions -				(B) slack	Docs Scout 🕐 🗸
ProjectPlanner.jo 🕲 explorer-ingest			© Las	st 6 hrs ending now 🗸	Compare To 🗸
Overview Web Endpoints V Background Jobs V	Database Alerts	Settings		_	Metadata Deploys
Q Filter Endpoints Showing first 50 most time	ne-consuming endpoints.	Show All (57)			
Name	Time Consumed V	Response Time	Throughput	Max Allocations	Error Rate
DeploysController#markers	22.3%	60 m	ns 41.9 rpm	110 K	0.0 rpm
AlertsController#list	22.1%	www.424 r	ms 5.9 rpm	3.1 M	0.0 rpm
StatusPagesController#status	17.6%	35 m	ns 56.0 rpm	22 K	0.0 rpm
AppsController#data	14.5%	279 r	ms 5.9 rpm	640 K	0.0 rpm
CheckinController#checkin	6.2%		s 47.0 rpm	28 K	0.0 rpm
AppsController#show	5.5%	280 r	ms 2.2 rpm	230 K	0.0 rpm
PublicController#index	3.3%	16 ms	s 22.9 rpm	19 K	0.0 rpm
Api::MetricsController#show	1.6%		ns 2.0 rpm	43 K	0.0 rpm
OrgController#billing	1.0%		ns 1.2 rpm	54 K	0.0 rpm
HealthCheckController#show	0.8%	78 m	ns 1.2 rpm	71 K	0.0 rpm
EndpointsController#show	0.8%	427 n	ms 0.2 rpm	180 K	0.0 rpm
AppsController#onboarding	0.4%	38 m	ns 1.3 rpm	20 K	0.0 rpm
EndpointsController#trace	0.4%	Manul_ml 525 r	ms 0.1 rpm	770 K	0.0 rpm
WorkersController#show	0.4%		ms 0.1 rpm	180 K	0.0 rpm
GitController#blame	0.4%	1,376	5 ms 0.0 rpm	48 K	0.0 rpm
EndpointsController#data	0.3%		ms 0.2 rpm	180 K	0.0 rpm
ServicesController#data	0.2%	1,317	ms 0.0 rpm	260 K	0.0 rpm
DevTracesController#show	0.2%	174 m	ns 0.1 rpm	180 K	0.0 rpm
EndpointsController#list	0.2%	1 1 1 mm 300 r	ms 0.1 rpm	700 K	0.0 rpm
GitController#annotate_backtrace	0.2%	677 m	ms 0.0 rpm	31 K	0.0 rpm
StripeController#webhook	0.2%	Λ_ΛΛ_Λ925 r	ms 0.0 rpm	36 K	0.0 rpm
Devise::SessionsController#new	0.2%	17 ms	s 1.0 rpm	11 K	0.0 rpm
WorkersController#index	0.1%	λ_ΛΛΛM\ 215 m	ms 0.1 rpm	160 K	0.0 rpm
EndpointsSparklinesController#index	0.1%		ns 0.1 rpm	120 K	0.0 rpm
ServicesController#index	0.1%	Δ. Δ. 562 π	ms 0.0 rpm	200 K	0.0 rpm
WorkersController#data	0.1%	~~ 176 m	ns 0.1 rpm	140 K	0.0 rpm

New Relic	Web	Endpoints
-----------	-----	-----------

New Relic.	~		Maps	Alerts ^{New} Tools ~ Help	o ∨ 🛛 🚧 app48563282@heroku	1 ~
Applications Service maps	Key transactions Alerts					
APPS APM	TIME PICKER Last 6 hours ending now All set	rvers 🗸				
MONITORING	Type Web \checkmark		Top 5 web transaction by percent of wall clock time	15 🕐 📐 th. %		
Overview Service maps	Most time consuming \checkmark		2000 %		. M	
Transactions	CheckinController#checkin	94.1%	1500 %		- And a Married	
Databases	CheckinController#app_server_load	4.46%	1000 %			\sim
External services	DeploysController#markers	0.34%	1000 %			
Instances	AlertsController#list	0.33%	500 %			
Ruby VMs	StatusPagesController#status	0.27%	9:00 AM 10	:00 AM 11:00 AM	12:00 PM 1:00 PM	2:00 PM
EVENTS	AppsController#data	0.26%	CheckinController#checkin	CheckinController#app_server_lo	ad DeploysController#markers	
Error analytics	AppsController#show	0.09%				
Errors	PublicController#index	0.05%	Throughput (rpm)			
Alerts	Api::MetricsController#show	0.03%	10k			
Deployments	OrgController#billing	0.02%	7.5k	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		~~~
Thread profiler	HealthCheckController#show	0.02%	5k			
	EndpointsController#show	0.01%	2500			
REPORTS	EndpointsController#trace	0.01%	0.000 AM	00 AM 11.00 AM	12.00 PM 1.00 PM	2.00 PM
SLA	AppsController#onboarding	0.01%	Non-Web Web	00 AM 11:00 AM	12:00 PM 1:00 PM	2:00 PM
Availability	EndpointsController#data	0.01%				
Capacity	WorkersController#show	0.01%	Transaction traces		Search traces	٩,
Web transactions	GitController#blame	0.01%	Date	Transaction / Details	^ Ann s	en/er ^
Database	DevTracesController#show	0%	Juic V	CheckinController#checkin	o vhh a	
Background jobs	StripeController#webhook	0%	12:13 — about 2 hours ago	/apps/checkin.scout	9,	,877 ms
background jobs	EndpointsController#list	0%	12:12 — about 2 hours ago	CheckinController#checkin	8,	,401 ms
SETTINGS	Show all transactions table			/apps/cneckin.scout		
Application			11:10 — about 3 hours ago	/apps/checkin.scout	7,	,806 ms
Availability monitoring Environment			11:07 — about 3 hours ago	CheckinController#checkin /apps/checkin.scout	7,	,909 ms
			10:15 — about 4 hours ago	CheckinController#checkin /apps/checkin.scout	8,	,603 ms
ALERIS			09:51 — about 4 hours ago	CheckinController#checkin /apps/checkin.scout	7,	,831 ms
			08:39 — about 6 hours ago	CheckinController#checkin /apps/checkin.scout	9,	,615 ms
					Show more slow trans	sactions

New Relic opts for several different visualizations on this page. Scout goes with a filterable, sortable view.

Both sort web endpoints by percent time consumed by default, helping you focus on the controller-actions that consume the greatest amount of time.

CHAPTER 5 Web Endpoint Detail

In the same way the app overview provides a breakdown of time spent, both New Relic and Scout offer endpoint-specific views for this:

Scout	Web Endpo	oint Detail		
\rightarrow				
Applications	÷		# slav	ck Docs Scout 🕐 🗸
ProjectPlanneric	D it explorer-intest		() Last 6 hrs and ing now	Compare To
Overview Web Endp	politis Background Jobs	Database Alerts Settings	O Last offics ending now	Metadata Deploys
CheckinContro	oller#checkin			Last Updated: 02:31PM
2010 2010 0 ex 0 ex 0 ex 0 ex 0 ex 0 ex 0 ex 0 e	9.00 am 9.30 am Throughput View AN RESPONSE TIME - 95TH 1,092	10.30 am 10.00	ActiveRecord Controller	3.000 rpm 3.000 rpm 5.000 rpm 5.000 rpm 5.000 rpm 5.000 rpm 5.000 rpm
269.5 ms	1,082 ms	2,686 / min 4M / req	U.U / min	0.9
Breakdown Showin	ng metrics across all transactions.			
Name	Time Consumed	Average Total Time	Calls Per-Transaction	
Controller	65.0%	165.9 ms	1.0	
InfluxDB	9.1%	23.2 ms	1.0	
Request#create	6.5%	16.7 ms	1.8	
Middleware	6.0%	15.2 ms	1.0	
Org#find	2.5%	6.3 ms	1.9	
InfluxRouter#find	2.1%	5.3 ms	1.9	
Hiding 11 metrics consuming a sm	nall percentage of time.			
Transaction Trac	CRDER BY: Slowest Response Time	\$		
When	URI	Duration	Allocations	Memory Increase
11:03:25 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1 27.0.01	13,445 ms	3.9 M	ОМВ
8:40:27 AM	/apps/checkin?agent=2.3.0&key=[KEY]	12,157 ms	3.8 M) MB
11:02:25 AM	/apps/checkin?agent=2.3.0&key=[KEY]	12,133 ms	3.7 M) MB
10:55:20 AM	/apps/checkin?agent=2.3.0&key=[KEY]	11,712 ms	3.7 M) MB
8:35:24 AM	/apps/checkin?agent=2.3.0&key=[KEY]	11,537 ms	3.9 M) MB
8:41:26 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1270.0.1	11,503 ms	4 M) MB
11:06:25 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1270.0.1	10,971 ms	3.7 M) MB
11:13:23 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1270.0.1	10,891 ms	3.8 M) MB
11:10:23 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1270.0.1	10,679 ms	3.6 M) MB
9:21:21 AM	/apps/checkin?agent=2.3.0&key=[KEY] 1 270.01	10,598 ms	3.8 M) MB

New Relic Web Endpoint Detail

\Rightarrow			
New Relic.	~		Maps Alerts ^{New} Tools ~ Help ~ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Applications Service maps	Key transactions Alerts		
APPS APM	TIME PICKER SERVERS Last 6 hours ending now All serv	vers 💛	
MONITORING	Type Web \checkmark		CheckinController#checkin (×)
Overview Service maps	Most time consuming $\qquad \lor$		Track as key transaction
Transactions	CheckinController#checkin	94.1%	App performance Historical performance Map ^{Beta}
Databases	CheckinController#app_server_load	4.46%	App server breakdown 📥 🐀 % 0.92 314 ms
External services	DeploysController#markers	0.34%	600 ms
Instances	AlertsController#list	0.33%	400 ms
Ruby VMs	StatusPagesController#status	0.27%	200 ms
EVENITO	AppsController#data	0.26%	
Even is	AppsController#show	0.09%	9:00 AM 10:00 AM 11:00 AM 12:00 PM 1:00 PM 2:00 PM
Errors	PublicController#index	0.05%	
Alerts	Api::MetricsController#show	0.03%	CheckInController#dheckin Apm::Application#call Net:HTTP[apm-production-Influx-a-4]: POST ActionDispatch::ParamsParser#call ActionDispatch::Routing::RouteSet#call Other
Deployments	OrgController#billing	0.02%	Throughput 2,730 rpm
Thread profiler	HealthCheckController#show	0.02%	3000 rpm AVERAGE
in coo promer	EndpointsController#show	0.01%	2000 rpm
REPORTS	EndpointsController#trace	0.01%	1000 rpm
SLA	AppsController#onboarding	0.01%	9:00 AM 10:00 AM 11:00 AM 12:00 PM 1:00 PM 2:00 PM
Availability	EndpointsController#data	0.01%	
Capacity	WorkersController#show	0.01%	Breakdown table
Scalability	GitController#blame	0.01%	Avg calls Avg Category Segment %Time (per txn) time (ms)
Web transactions	DevTracesController#show	0%	Controller CheckinController#checkin 54.6 1.0 171
Database	StripeController#webbook	0%	Pack App Apprication#call 14.4 1.0 45.2
Background jobs	EndpointsController#list	0%	External Net: HTTP[app production influx a 4]: POST 71 197 22 2
SETTINGS	Show all transactions table	0.0	Middlawara ArtionDispatch=DarameDarcer#coll 2.5 4.0 44
Application			Middleware ActionDispatchuralation20uteCetteral 2.0 4.0 0.45
Availability monitoring			minuneware ActionDispatchcoutingcouteset#Call 3.0 1.0 9.45
Environment			Database Postgres other 2.4 1.84 7.66
			Database Postgres Request insert 2.1 1.75 6.66
ALERTS >			Database Redis multi 1.7 3.11 5.18
			Show all segments →
			Transaction traces Sample performance details
			Date University of the Transaction / Details
			12:13 — about 2 hours ago /apps/checkin.scout 9,877 ms

There are some differences on this page:

- New Relic offers a histogram display option of response times and Scout does not.
- Scout offers memory allocation metrics (more allocations lead to increased memory usage) while New Relic does not.
- New Relic's breakdown provides more categories, while Scout focuses the breakdown on database queries.
- Scout typically provides more transaction traces (up to 10 per-minute) and more options for sorting and selecting traces.

CHAPTER 6 Database Monitoring

The database is the most common bottleneck for web applications. Since a database is a shared resource, one expensive query may cause many other types of queries to run slower. Both tools provide additional analytics around database query performance:

Sc	out Database	Monitor	ing			
** -	Applications					🕲 slack Docs Scout 🔱 🗸
Project	Planner.io و explorer-ingest				C Last 60 minutes er	nding now 👻 Compare To 🗸
Overview	Web Endpoints 👻 Background Jobs 👻	Database Alerts	Settings			Metadata Deploys
Time Cons	sumed by Query vs. Throughput					Database Events
30 ms	h		~~~		6,000 rpm	USAGE SPIKE Time spent in database queries spiked 114% O <u>Sun 2:00 AM</u> - about 11 hours ago
19 n s					2.00 cmm	Time spent in database queries spiked 34% Standard Standard St Standard Standard
	12:30 pm 12:35 pm 12:40 pm 12:45 pm	12:50 pm 12:55 pm 1:0	00 pm 1:05 pm 1	:10 pm 1:15 p	om 1:20 pm	USAGE SPIKE
	- Inrougnput SlowJob#save Provis	ioning::Herokuinstali#tind	rg#tina 📕 kequest#tir	id Uther Qi	ueries	26%
Queries Li	ist					
				QUERY TIME		USAGE SPIKE Time spent in database queries spiked
Rank	Name	% Time Consumed	Throughput	Mean	95th	23% © Sun 12:25 PM + about 1 hour ago
1	Request#find via TraceAnalysisJob	20.6%	1,259.6 /min	2.4 ms	4.7 ms	USAGE SPIKE
2	SlowJob#save via JobTraceAnalysisJob	10.2%	324.2 /min	4.7 ms	9.6 ms	Time spent in database queries spiked 22%
3	Provisioning::Herokulnstall#find via PlanTransactionsEnforcementJob	10.2%	653.1 /min	2.3 ms	4.8 ms	Sun 8:30 AM • about 5 hours ago
4	Provisioning::ManifoldInstall#find via PlanTransactionsEnforcementJob	9.1%	653.1 /min	2.1 ms	4.6 ms	
5	Org#find via PlantransactionsEnforcementJob	9.0%	653.1 /min	2.0 ms	3.9 ms	
6	SlowJob#find vis JobTraceAnalysisJob	5.2%	324.2 /min	2.4 ms	4.7 ms	
7	FlaggedTrace#create via TraceAnalysisJob	3.5%	38.0 /min	13.7 ms	36.0 ms	
8	Deploy#find via DeploysController#markers	2.4%	70.7 /min	5.0 ms	9.1 ms	
9	NodeDeploy#find via MarkPreviousNodeDeployJob	2.2%	2.6 /min	127.9 ms	280.9 ms	
10	FlaggedTrace#create via JobTraceAnalysisJob	2.0%	23.6 /min	12.7 ms	30.4 ms	

New Re	elic Database	e Mo	onitoring
<>			
	~		Maps Alerts ^{New} Tools V Help V 🛹 app48563282@heroku 🔳 V
Applications Service maps	Key transactions Alerts		
APPS APM ~	TIME PICKER Last 6 hours ending now All serv	ers 🗸	
MONITORING Overview Service maps Transactions	DATABASE All Postgres ActiveRecord SORT BY Most time consuming Postgree Request insert	Redis	Postgres overview Top database operations by time consumed 125 100 75
External services	Postgres Org find	17.2 s	
Instances	Postgres Request find	11.2 s	
Ruby VMs	Postgres InfluxServer find	11 s	0
EVENTS	Postgres InfluxRouter find	11 s	8:00 AM 9:00 AM 10:00 AM 11:00 AM 12:00 PM 1:00 PM Postgres Org find Postgres InfluxServer find Postgres InfluxServer find Postgres InfluxServer find
Error analytics	Postgres NodeDeploy find	9.5 s	Postgres Request find
Errors	Postgres App find	8.68 s	
Alerts	Postgres Node find	6.78 s	Top database operations by query time
Deployments	Postgres AppMetadata find	6.29 s	25 ms
Thread profiler	Postgres Provisioning::Herokulnstall f	5.73 s	20 ms
	Postgres SlowJob other	5.52 s	15 ms
REPORTS	Postgres Node update	5.51 s	10 ms
SLA	Postgres Provisioning::ManifoldInstall	5.23 s	5 ms
Availability	Postgres SlowJob insert	4.38 s	8:00 AM 9:00 AM 10:00 AM 11:00 AM 12:00 PM 1:00 PM
Capacity	Postgres SlowJob find	2.8 s	Postgres insert Postgres other Postgres find Postgres select Postgres update Postgres show
Scalability	Postgres FlaggedTrace other	2.46 s	
Web transactions	Postgres AppMetadata other	1.6 s	Top database operations by throughput
Database	Postgres FlaggedTrace insert	0.609 s	60k cpm
Background Jobs	Postgres Deploy find	0.438 s	
SETTINGS	Postgres AlertCondition find	0.373 s	40k cpm ~~~ lun ~~ lun
Application	Show all database operations table		30k cpm
Availability monitoring			zuk opm
Environment			

Both of the UIs resemble each tool's web endpoints display but made specifically for database queries. There are some subtle yet important differences between the two:

- Scout's chart displays data across every query not just the top five most expensive queries by combining less expensive queries into an "other queries" metric. This gives a more complete picture of performance.
- New Relic provides monitoring beyond ActiveRecord. Scout focuses on ActiveRecord.

• Scout's list of queries includes the calling controller-action or background job. Expensive queriers are frequently triggered by a few hotspots in the code: this makes it easier to identify those spots at a glance. Finally, Scout makes it easy to compare performance in a small slice of time to the normal performance. This isn't possible with New Relic - you can change the entire timezone, but you can't compare between a small slice of time and a larger one. Here's how this looks within Scout:



CHAPTER 7 Background Jobs

Both New Relic and Scout monitor the core Ruby background job frameworks: Sidekiq, Delayed Job, and Resque. The UI is similar to endpoints, so I haven't provided a screenshot here.

CHAPTER 8Development Profiler

When you are actively resolving a performance issue, it's helpful to get immediate feedback on the results as you code. Otherwise, you need to deploy and wait for fresh data to verify the change had the intended impact.

New Relic used to have a developer mode, but it has since been removed from the agent.

Scout DevTrace gives immediate access to traces in your local development environment by clicking on a speed badge in the lower left of your browser:

Scout	Development Profiler
New Customers	s ~ Los Pollos Hermanos
Jesse Pinkman	
Denver, CO	
Denier, OO	
345 ms	

Slow feedback cycles - like a long-running test suite - are painful. DevTrace eliminates a lot of this as you can instantly verify that an optimization - like fixing an N+1 query with `includes` - is working as expected before deploying.

chapter 9 Weekly Trends Email

A recurring report of your app's health is a great way for managers to stay in the loop on performance and to uncover slowly building problems. New Relic puts more emphasis on high-level numbers. Scout digs more into the source of problems and trends:

Scou	t Wee	kly Em	ail			
	Scout 🔫 👘					
	Across 5 apps, web prior week. We've highlighted Pr	Were there any deploys				
Summary of web & background job performance for the app.	ProjectPlanner.io	yesterday?				
	WEB Response Time	Throughput	BACKGROUND JOBS Execution Time	Throughput	Insignificant changes are faded for less	
	↓10% 50 ms	16% 500 /min	↓10% 25 ms	↓2% 1,000 /min	attention.	
	Response times we call times increas	Drills deep into app performance, looking for				
Ties performance trends to a specific deploy, if possible.	UsersCo Performance	specific categories of metrics with slower response times.				
Identifies endpoints that are triggering an abnormal number of	PhotosC 7 second					
slow requests.	A single Pro	Calls out the source of the bottleneck.				

New Relic Weekly Email

🔘 New Reli	c . _{in}	partn	ershi	io witl	h Her	oku A	Add-O	ns						
Monday	me	etri	cs	fo	r a	pp	48	5632	82@	he	roku.com			
Because Mondays need a little help. Brought to you by Heroku Add-Ons.														
Monday 10/02/2017 three	ough S	Sunday	10/08/	2017										
Skip ahead to all apps r	eport													
ProjectPla	nne	ər.ic)										See report a	at New Relic
1.71 sec Page load time				23 Viev	.2к vs			0.99 Apdex			0.0022 % Error rate			
▼ 6.56 %			▼ 3.73 %			no change			▲ 29.4 %	From last	From last week			
▼ 34.7 %		▼ 9.02 %		▲ 3.13 %			▼ 55.1 %	From 12	weeks ago					
Two month	n th	irou	igh	ipu	t pa	atte	ern							
								0.5	73 ppm					
August 2017	S	Μ	т	W	т	F	S			Quie	test day			
										Busi	est day			
September 2017														

CHAPTER 10

Agent, Alerting & Error Analytics

Both the New Relic and Scout agents are battle-tested. In <u>Scout's</u> <u>open-sourced benchmarks</u>, **Scout's agent was shown to have lower overhead**:

APM Agent	Response Time (Mean)	Response Time (95th Percentile)	Response Time (Max)	Overhead	
None	55.6 ms	106.4 ms	2,174.1 ms		
New Relic	80.4 ms	149.5 ms	2,263.5 ms	44.5%	
Scout APM	56.8 ms	102.7 ms	2,168.7 ms	2.2%	

The impact of agent overhead increases as CPU resources on your app servers becomes more scarce.

Scout and New Relic have comparable **alerting functionality**.

Scout tracks error rates and integrates with third-party services like Rollbar to provide details on exceptions. New Relic can provide this data, including backtraces to exceptions.

Conclusion

New Relic and Scout require little configuration - outside adding a dependency - to try. You can run Scout and New Relic at the same time without causing conflicts.

Our suggestion? <u>Try Scout's free 14-day trial</u> alongside New Relic, then ask: "which tool helps your team solve performance issues faster?"